Black Hole Search with Finite Automata Scattered in a Synchronous Torus

Jérémie Chalopin[1], Shantanu Das[1], Arnaud Labourel[1], and Euripides Markou[2]

[1]LIF, Aix-Marseille University, Marseille, France.
Email: {jeremie.chalopin,shantanu.das,arnaud.labourel}@lif.univ-mrs.fr
[2]Department of Computer Science and Biomedical Informatics,
University of Central Greece, Lamia, Greece. Email: emarkou@ucg.gr

Abstract

We consider the problem of locating a black hole in synchronous anonymous networks using finite state agents. A black hole is a harmful node in the network that destroys any agent visiting that node without leaving any trace. The objective is to locate the black hole without destroying too many agents. This is difficult to achieve when the agents are initially scattered in the network and are unaware of the location of each other. In contrast to previous results, we solve the problem using a small team of finite-state agents each carrying a constant number of identical tokens that could be placed on the nodes of the network. Thus, all resources used in our algorithms are independent of the network size.

We restrict our attention to oriented torus networks and first show that no finite team of finite state agents can solve the problem in such networks, when the tokens are not movable, i.e., they cannot be moved by the agents once they have been released on a node. In case the agents are equipped with movable tokens, we determine lower bounds on the number of agents and tokens required for solving the problem in torus networks of arbitrary size. Further, we present a deterministic solution to the black hole search problem for oriented torus networks, using the minimum number of agents and tokens, thus providing matching upper bounds for the problem.

Keywords: Distributed Algorithms, Fault Tolerance, Black Hole Search, Anonymous Networks, Mobile Agents, Identical Tokens, Finite State Automata

1 Introduction

The exploration of an unknown graph by one or more mobile agents is a classical problem initially formulated in 1951 by Shannon [28] and it has been extensively studied since then (e.g., see [1, 9, 21]). Recently, the exploration problem has also been studied in unsafe networks which contain malicious hosts of a highly harmful nature, called *black holes*. A black hole is a node which contains a stationary process destroying all mobile agents visiting this node, without leaving any trace. In the *Black Hole Search* problem the goal for the agents is to locate the black hole within finite time. In particular, at least one agent has to survive knowing all edges leading to the black hole. The only way of locating a black hole is to have at least one agent visiting it. However, since any agent visiting a black hole is destroyed without leaving any trace, the location of the black hole must be deduced by some communication mechanism employed by the agents. Four such mechanisms have been proposed in the literature: a) the *whiteboard* model in which there is a whiteboard at each node of the network where the agents can leave messages, b) the *'pure' token* model where the agents carry tokens which they can leave at nodes, c) the *'enhanced' token* model in which the agents can leave tokens at nodes or edges, and d) the time-out mechanism (only for synchronous networks) in which one agent explores a new node while another agent waits for it at a safe node.

The most powerful inter-agent communication mechanism is having whiteboards at all nodes. Since access to a whiteboard is provided in mutual exclusion, this model could also provide the agents a symmetry-breaking mechanism: If the agents start at the same node, they can get distinct identities and then the distinct agents can assign different labels to all nodes. Hence in this model, if the agents are initially co-located, both the agents and the nodes can be assumed to be non-anonymous without any loss of generality.

In asynchronous networks and given that all agents initially start at the same safe node, the Black Hole Search (BHS) problem has been studied under the whiteboard model (e.g., [10, 13, 14, 15]), the 'enhanced' token model (e.g., [11, 16, 29]) and the 'pure' token model in [19]. It has been proved that the problem can be solved with a minimal number of agents performing a polynomial number of moves. Notice that in an asynchronous network the number of the nodes of the network must be known to the agents otherwise the problem is unsolvable ([14]). If the graph topology is unknown, at least $\Delta + 1$ agents are needed, where Δ is the maximum node degree in the graph ([13]). Furthermore the network should be 2-connected. It is also not possible to answer the question of *whether* a black hole exists in the network.

In asynchronous networks, with scattered agents (not initially located at the same node), the problem has been investigated for the ring topology ([12, 14]) and for arbitrary topologies ([20, 3]) in the whiteboard model while in the 'enhanced' token model it has been studied for rings ([17, 18]) and for some interconnected networks ([29]).

The consideration of synchronous networks makes a dramatic change to the problem. Now two co-located distinct agents can discover one black hole in any graph by using the time-out mechanism, without the need of whiteboards or tokens. Moreover, it is now possible to answer the question of whether a black hole actually exists or not in the network. No knowledge about the number of nodes is needed. Hence, with co-located distinct agents, the issue is not the feasibility but the time efficiency of black hole search. The issue of efficient black hole search has been studied in synchronous networks without whiteboards or tokens (only using the time-out mechanism) in [4, 5, 7, 8, 23, 24, 25] under the condition that all distinct agents start at the same node. However when the agents are scattered in the network, the time-out mechanism is not sufficient anymore.

Indeed the problem seems to be much more difficult in the case of scattered agents and there are very few known results for this scenario. In this paper we study this version of the problem using

very simple agents that can be modeled as finite state automata. Our objective is to determine the minimum resources, such as number of agents and tokens, necessary and sufficient to solve the problem in a given class of networks. For the class of ring networks, recent results [2] show that having constant-size memory is not a limitation for the agents when solving this problem. We consider here the more challenging scenario of anonymous torus networks of arbitrary size. We show that even in this case, finite state agents are capable of locating the black hole in all oriented torus networks using only a few tokens. Note that the exploration of anonymous oriented torus networks is a challenging problem in itself, in the presence of multiple identical agents [27]. Since the tokens used by the agents are identical, an agent cannot distinguish its tokens from those of another agent.

While the token model has been mostly used in the exploration of safe networks, the whiteboard model is commonly used in unsafe networks. The 'pure' token model can be implemented with $O(1)$-bit whiteboards, for a constant number of agents and a constant number of tokens, while the 'enhanced' token model can be implemented having a $O(\log d)$-bit whiteboard on a node with degree d. In the whiteboard model, the capacity of each whiteboard is always assumed to be of at least $\Omega(\log n)$ bits, where n is the number of nodes of the network. In all previous papers studying the Black Hole Search problem under a token model apart from [19] and [2], the authors have used the 'enhanced' token model with agents having non-constant memory. The weakest 'pure' token model has been used in [19] for co-located non-constant memory agents equipped with a map in asynchronous networks.

The Black Hole Search problem has also been studied for co-located agents in asynchronous and synchronous directed graphs with whiteboards in [6, 25]. In [22] they study the problem in asynchronous networks with whiteboards and co-located agents without the knowledge of incoming link. A different dangerous behavior is studied for co-located agents in [26], where the authors consider a ring and assume black holes with Byzantine behavior, which do not always destroy a visiting agent.

Our Contributions: We consider the problem of locating the black hole in an anonymous but oriented torus containing exactly one black hole, using a team of identical agents that are initially scattered within the torus. Henceforth we will refer to this problem as the BHS problem. We focus our attention on very simple mobile agents. The agents have constant-size memory, they can communicate with other agents only when they meet at the same node and they carry a constant number of identical tokens which can be placed at nodes. The tokens may be movable (i.e. they can be released and picked up later) or unmovable (i.e. they cannot be moved by the agents once they have been released on a node). We prove the following results:

- No finite team of agents can solve the BHS problem in all oriented torus networks using a finite number of *unmovable* tokens.
- For agents carrying any finite number of *movable* tokens, at least three agents are required to solve the problem.
- Any algorithm for solving BHS using 3 agents requires more than one movable token per agent.
- The BHS problem can be solved using three agents and only two movable tokens per agent, thus matching both the lower bounds mentioned above.

In Section 2, we formally describe our model, giving the capabilities of the agents. In Section 3, we prove lower bound on the number of agents and tokens needed to solve the BHS problem in

the torus. In Section 4, we present two deterministic algorithms for BHS: (i) using $k \geq 3$ agents carrying 3 movable tokens per agents, and (ii) using $k \geq 4$ agents carrying 2 movable tokens per agent. In Section 5, we present a more involved algorithm that uses exactly 3 agents and 2 tokens per agent thus meeting the lower bounds. All our algorithms are time-optimal and since they do not require any knowledge about the dimensions of the torus, they work in any synchronous oriented torus, using only a finite number of agents having constant-size memory.

2 Our Model

Our model consists of $k \geq 2$ anonymous and identical mobile agents that are initially placed at distinct nodes of an anonymous, synchronous torus network of size $n \times m$, $n \geq 3$, $m \geq 3$. We assume that the torus is oriented, i.e., at each node, the four incident edges are consistently marked as North, East, South and West. Each mobile agent owns a constant number of t identical tokens which can be placed at any node visited by the agent. In all our protocols a node may contain at most three tokens at the same time and an agent carries at most three tokens at any time. A token or an agent at a given node is visible to all agents on the same node, but is not visible to any other agent. The agents follow the same deterministic algorithm and begin execution at the same time and being at the same initial state.

At any single time unit, a mobile agent occupies a node u of the network and may 1) detect the presence of one or more tokens and/or agents at node u, 2) release/take one or more tokens to/from the node u, and 3) decide to stay at the same node or move to an adjacent node. We call a token *movable* if it can be moved by any mobile agent to any node of the network, otherwise we call the token *unmovable* in the sense that, once released, it can occupy only the node in which it has been released.

Formally we consider a mobile agent as a finite Moore automaton $\mathcal{A} = (\mathcal{S}, S_0, \Sigma, \Lambda, \delta, \phi)$, where $\mathcal{S}$ is a set of $\sigma \geq 2$ states; S_0 is the *initial* state; Σ is the set of possible configurations an agent can see when it enters a node; $\delta : \mathcal{S} \times \Sigma \rightarrow \mathcal{S}$ is the transition function; and $\phi : \mathcal{S} \rightarrow \Lambda$ is the output function. Elements of Σ are quadruplets (D, x, y, b) where $D \in \{\texttt{North}, \texttt{South}, \texttt{East}, \texttt{West}, \texttt{none}\}$ is the direction through which the agent has arrived at the node, x is the number of tokens (at most 3) at that node, y is number of tokens (at most 3) carried by the agent and $b \in \{true, false\}$ indicates whether there is at least another agent at the node or not. Elements of Λ are quadruplets (P, s, X, M) where $P \in \{\texttt{put}, \texttt{pick}\}$ is the action performed by the agent on the tokens, $s \in \{0, 1, 2, 3\}$ is the number of tokens concerned by the action A, $X \in \{\texttt{North}, \texttt{South}, \texttt{East}, \texttt{West}, \texttt{none}\}$ is the edge marked as dangerous by the agent, and $M \in \{\texttt{North}, \texttt{South}, \texttt{East}, \texttt{West}, \texttt{none}\}$ is the move performed by the agent. Note that the agent always performs the action before the marking and the marking before the move.

Note that all computations by the agents are independent of the size $n \times m$ of the network since the agents have no knowledge of n or m. There is exactly one black hole in the network. An agent can start from any node other than the black hole and no two agents are initially co-located. Once an agent detects a link to the black hole, it marks the link permanently as dangerous (i.e., disables this link). Since the agents do not have enough memory to remember the location of the black hole, we require that at the end of a black hole search scheme, all links incident to the black hole (and only those links) are marked dangerous and that there is at least one surviving agent. Thus, our definition of a successful BHS scheme is slightly different from the original definition. The time complexity of a BHS scheme is the number of time units needed for completion of the scheme, assuming the worst-case location of the black hole and the worst-case initial placement of the scattered agents.

3 Impossibility results

In this section we give lower bounds on the number of agents and the number and type of tokens needed for solving the BHS problem in any anonymous, synchronous and oriented torus.

3.1 Agents with unmovable tokens

We will prove that any constant number of agents carrying a constant number of unmovable tokens each, can not solve the BHS problem in an oriented torus. The idea of the proof is the following: We show that an adversary (by looking at the transition function of an agent) can always select a big enough torus and initially place the agents so that no agent visits nodes which contain tokens left by another agent, or meets with another agent. Moreover there are nodes on the torus never visited by any agent. Hence the adversary may place the black hole at a node not visited by any of the agents to make the algorithm fail.

Theorem 3.1 *For any constant numbers k, t, there exists no algorithm that solves BHS in all oriented tori containing one black hole and k scattered agents, where each agent has a constant memory and t unmovable tokens.*

To prove Theorem 3.1 we will need the following two propositions which appeared in [27].

Proposition 3.1 *[27] Consider one mobile agent with σ states and a constant number t of identical unmovable tokens. We can always (for any configuration of the automaton, i.e., states and transition function) select a $n \times n$ oriented torus, where $n > t\sigma^2$ so that no matter what is the starting position of the agent, it cannot visit all nodes of the torus. In fact, the agent will visit at most $\sigma + t(\sigma-1)^2(n+1) < n^2$ nodes.*

Proposition 3.2 *[27] Let A be an agent with σ states and a constant number t of identical unmovable tokens in a $n \times n$ oriented torus, where $n > t\sigma^2$ and let v be a node in that torus. There are at most $\sigma + t(\sigma-1)^2(n+1) < n^2$ different starting nodes that we could have initially placed A so that node v is always visited by A.*

Proof of Theorem 3.1: Consider a constant number of k mobile agents with σ states and a constant number of t unmovable tokens each, in an $n \times n$ oriented torus. We show that an adversary can always (for any configuration of the automatons, i.e., states and transition function) initially place the agents on the torus and select n so that there are nodes on the torus never visited by any agent.

Take a $n \times n$ oriented torus, where $n > 2kt^2\sigma^2$ and let $s(A_1)$ be the starting node of agent A_1. If agent A_1 was alone in the torus would release its tokens at nodes $T_1(A_1), T_2(A_1), \ldots, T_t(A_1)$. According to Proposition 3.2, there are at least $n^2 - (\sigma + t(\sigma-1)^2(n+1))$ starting nodes at which an adversary could place agent A_2 so that A_2 does not visit node $T_1(A_1)$. Among these starting nodes (applying again Proposition 3.2) there are at most $\sigma + t(\sigma-1)^2(n+1)$ nodes that would lead agent A_2 to token $T_2(A_1)$, another at most $\sigma + t(\sigma-1)^2(n+1)$ nodes that would lead agent A_2 to token $T_3(A_1)$ and so on. Therefore there are at least $n^2 - t(\sigma + t(\sigma-1)^2(n+1))$ starting nodes at which the adversary could place agent A_2 so that A_2 does not visit any of the $T_1(A_1), T_2(A_1), \ldots, T_t(A_1)$ nodes. The adversary still needs to place agent A_2 at a starting node $s(A_2)$ so that A_2 releases its tokens at nodes $T_1(A_2), T_2(A_2), \ldots, T_t(A_2)$ not visited by agent A_1.

Notice that an agent can decide to release a new token at a distance of at most σ nodes from a previously released token (an agent cannot count more than σ before it repeats a state).

Since $n > 2kt^2\sigma^2$ for every two different starting nodes $s(A_2)$ and $s'(A_2)$, agent A_2 would release its tokens to nodes $T_1(A_2), T_2(A_2), \ldots, T_t(A_2)$ and $T'_1(A_2), T'_2(A_2), \ldots, T'_t(A_2)$ respectively, where $T_i(A_2) \neq T'_i(A_2), 1 \leq i \leq t$.

Since in view of Proposition 3.1 agent A_1 can visit at most $\sigma + t(\sigma-1)^2(n+1)$ nodes (if A_1 was alone in the torus), there are at most $\sigma + t(\sigma-1)^2(n+1)$ starting nodes for A_2 for which A_2 would place its first token at a node visited by agent A_1, another at most $\sigma + t(\sigma-1)^2(n+1)$ starting nodes for A_2 for which A_2 would place its second token at a node visited by agent A_1, and so on. Hence we need to exclude another $t(\sigma + t(\sigma-1)^2(n+1))$ starting nodes for agent A_2. Thus we have left with $n^2 - 2t(\sigma + t(\sigma-1)^2(n+1))$ starting nodes at which the adversary can place agent A_2 so that A_2 does not visit any of the $T_1(A_1), T_2(A_1), \ldots, T_t(A_1)$ nodes and agent A_1 does not visit any of the $T_1(A_2), T_2(A_2), \ldots, T_t(A_2)$ nodes.

For the placement of agent A_3, following the same reasoning, and using again Proposition 3.2, we have that there are at least $n^2 - 2t(\sigma + t(\sigma-1)^2(n+1))$ starting nodes at which the adversary could place agent A_3 so that A_3 does not visit any of the $T_1(A_1), T_2(A_1), \ldots, T_t(A_1)$ or $T_1(A_2), T_2(A_2), \ldots, T_t(A_2)$ nodes. And using Proposition 3.1 as above by excluding another $2t(\sigma + t(\sigma-1)^2(n+1))$ starting nodes for agent A_3, we have left with $n^2 - 4t(\sigma + t(\sigma-1)^2(n+1))$ starting nodes at which the adversary can place agent A_3 so that A_3 does not visit any of the $T_1(A_1), T_2(A_1), \ldots, T_t(A_1)$ or $T_1(A_2), T_2(A_2), \ldots, T_t(A_2)$ nodes and agents A_1 and A_2 do not visit any of the $T_1(A_3), T_2(A_3), \ldots, T_t(A_3)$ nodes.

Following the same reasoning, an adversary can select a node out of

$$n^2 - 2(k-1)t(\sigma + t(\sigma-1)^2(n+1))$$

nodes to place agent A_k so that A_k does not visit any of the $T_1(A_j), T_2(A_j), \ldots, T_t(A_j)$ nodes, where $1 \leq j \leq k-1$, and agents A_j do not visit any of the $T_1(A_k), T_2(A_k), \ldots, T_t(A_k)$ nodes.

Hence all agents may only visit their own tokens and they have to do it at the same time and being at the same states and therefore they maintain their initial distance forever. Since any agent may only see its own tokens, by Proposition 3.1, any agent can visit at most $\sigma + t(\sigma-1)^2(n+1)$ nodes on the torus and hence all agents will visit at most $k(\sigma + t(\sigma-1)^2(n+1)) < n^2$ nodes when $n > 2kt^2\sigma^2$. Therefore there are nodes never visited by any agent and the adversary can place the black hole at such a node. ■

3.2 Agents with movable tokens

We first show that the BHS problem is unsolvable in any synchronous torus by two scattered agents having any number of movable tokens even if the agents have unlimited memory.

Lemma 3.1 *Two agents carrying any number of movable tokens cannot solve the BHS problem in an oriented torus even if the agents have unlimited memory.*

Proof : Assume w.l.o.g. that the first move of the agents is going East. Suppose that the black hole has been placed by an adversary at the East neighbor of an agent. This agent vanishes into the black hole after its first move. The adversary places the second agent such that it vanishes into the black hole after its first vertical move, or it is in a horizontal ring not containing the black hole if the agent never performs vertical moves. Observe that the trajectories of the two agents intersect only at the black hole and neither can see any token left by the other agent. Neither of the agents will ever visit the East neighbor of the black hole and thus, they will not be able to correctly mark all links incident to the black hole. ■

Thus, at least three agents are needed to solve the problem. We now determine a lower bound on the number of tokens needed by three scattered agents to solve BHS.

Lemma 3.2 *There exists no universal algorithm that could solve the BHS problem in all oriented tori using three agents with constant memory and one movable token each.*

Proof : Clearly, in view of Theorem 3.1, an algorithm which does not instruct an agent to leave its token at a node, cannot solve the BHS problem. Hence any potentially correct algorithm should instruct an agent to leave its token down. Moreover this decision has to be taken after a finite number of steps (due to agents' constant memory). After that the agents visit new nodes until they see a token. Following a similar reasoning as in Theorem 3.1 we can show that if the agents visit only a constant number of nodes before returning to meet their tokens they cannot visit all nodes of the torus. If they move their tokens each time they see them and repeat the previous procedure (i.e., visit a constant number of nodes and return to meet their tokens), we can show that they will find themselves back at their initial locations and initial states without having met with other agents and leaving some nodes unvisited. An adversary may place the black hole at an unvisited node to make the algorithm fail. Now consider the case that at some point an algorithm instructs the agents to visit a non-constant number of nodes until they see a token (e.g., leave your token down and go east until you see a token). Again in a similar reasoning as in Theorem 3.1, we can show that an adversary may initially place the agents and the black hole, and select the size of the torus so that two of the agents enter the black hole without leaving their tokens close to it: The agent (say A) that enters first into the black hole has been initially placed by an adversary so that it left its token more than a constant number of nodes away from the black hole. The adversary initially places another agent B so that it enters the black hole before it meets A's token. Furthermore B leaves its token more than a constant number of nodes away from the black hole. Hence the third agent, even if it meets the tokens left by A or B, it could not decide the exact location of the black hole. ■

We can prove using Lemma 3.2 and 3.1, the following theorem.

Theorem 3.2 *At least three agents are necessary to solve the BHS problem in an oriented torus of arbitrary size. Any algorithm solving this problem using three agents requires at least two movable tokens per agent.*

4 Simple Algorithms for BHS in a torus using moveable tokens

Due to the impossibility result from the previous section, we know that unmoveable tokens are not sufficient to solve BHS in a torus. In the following, we will use only moveable tokens. To explore the torus an agent uses the Cautious-Walk technique [14] using moveable tokens. In our algorithms, a *Cautious-Walk in direction D with x tokens* means the following: (i) the agent releases a sufficient number of tokens such that there are exactly x tokens at the current node, (ii) the agent moves one step in direction D and if it survives, the agent immediately returns to the previous node, (iii) the agent picks up the tokens released in step (i) and again goes one step in direction D. If an agent vanishes during step (ii), any other agent arriving at the same location sees x tokens and realizes a potential danger in direction D. Depending on the algorithm an agent may use 1, 2, or 3 tokens to implement the Cautious-Walk.

4.1 Solving BHS using $k \geq 3$ agents and 3 tokens

We show that three agents suffice to locate the black hole if the agents are provided with three tokens. We present an algorithm (BHS-torus-33) that achieves this. This algorithm uses three procedures : *Mark_All*, *Wait* and *Cautious-walk*.

The procedure *Mark_All*(D) is executed by an agent when it deduces that the next node w in the direction D from current location is the black hole. The agent traverses a cycle around node w, visiting each neighbor of w and marking as "dangerous" all links leading to node w.
The subroutine *Wait*(x) requires the agent to take no action for x time units.

Procedure Cautious-walk(Direction D, integer x)

```
   /* Procedure used by the agent to explore the nodes                                   */
 1 Put tokens until there are x tokens;
 2 Go one step along D and then go back;   /* test if the node in direction D is the black hole */
 3 Pick up the tokens released in the first step;
 4 Go one step along D;
```

Algorithm 1: BHS-Torus-33

```
   /* Algorithm for BHS in Oriented Torus (using k=3 or more agents, 3 tokens each)     */
   /* One token = Homebase                                                              */
   /* Two tokens = BlackHole in the East                                                */
   /* Three tokens = BlackHole in the South                                             */
 1
 2 Found:= false;
 3 while not Found do
 4     Count := 0;
 5     Put two tokens;
 6     Go one step East and come back;
 7     Pick one token and go one step East ;                  /* leaving one token at the homebase */
 8     repeat
 9         if found single token then increment Count;
10         CautiousWalk(East,2);
11     until found two tokens or count = 2;
12     if found 2 or 3 tokens then Found:=true;
13     else                /* The agent found 1 token and must move to the next horizontal ring */
14         Pick one token ;                                          /* Remove the homebase token */
15         CautiousWalk(South,3) ;                          /* using the token at the current node */
16         while found one token do            /* Current node is the homebase of another agent */
17             CautiousWalk(East,2) ;                       /* using the token at the current node */
18         if found 2 or 3 tokens then Found:=true;
19 if found 2 tokens then Mark-All(East);
20 if found 3 tokens then Mark-All(South);
```

Algorithm BHS-torus-33:
An agent explores one horizontal ring at a time and then moves one step South to the next horizontal ring and so on. When exploring a horizontal ring, the agent leaves one token on the starting node. This node is called the *homebase* of the agent and the token left (called homebase token) is used by the agent to decide when to proceed to the next ring. The agent then uses the two remaining tokens to repeat Cautious-Walk in the East direction until it has seen twice a node containing one token. Any node containing one token is a homebase either of this agent or of another agent. The agent moves to the next horizontal ring below after encountering two homebases. However before moving to the next ring, it does a cautious walk in the South direction with three tokens (the two tokens it carries plus the homebase token). If the agent survives and the node reached by the agent has one token, the agent repeats a cautious walk in the East direction (with two tokens) until it reaches an empty node. The agent can now use this empty

node as its new homebase. It then repeats the same exploration process for this new ring leaving one token at its new homebase.

Whenever the agent sees two or three tokens at the end of a cautious-walk, the agent has detected the location of the black hole: If there are two (resp. three) tokens at the current node, the black hole is the neighboring node w to the East (resp. South). In this case, the agent stops its normal execution and then traverses a cycle around node w, visiting all neighbors of w and marking all the links leading to w as dangerous.

Theorem 4.1 *Algorithm BHS-torus-33 correctly solves the BHS problem with* 3 *or more agents.*

Proof : An agent may visit an unexplored node only while going East or South. If one agent enters the black hole going East (resp. South), there will be two (resp. three) tokens on the previous node and thus, no other agent would enter the black hole through the same link. This implies that at least one agent always survives. Whenever an agent encounters two or three tokens at the end of a Cautious-Walk, the agent is certain of the location of the black hole since any alive agent would have picked up its Cautious-Walk tokens in the second step of the cautious walk (The agents move synchronously always using cautious walk and taking three steps for each move). ■

4.2 BHS using $k \geq 4$ agents and 2 tokens each

We now present an algorithm that uses only two tokens per agent, but requires at least 4 agents to solve BHS.

During the algorithm, the agents put two tokens on a node u to signal that either the black hole is on the South or the East of node u. Eventually, both the North neighbor and the West neighbor of the black hole are marked with two tokens. Whenever there is a node v such that there are exactly two tokens at both the West neighbor of v and the North neighbor of v, then we say that there exists a *Black-Hole-Configuration* (BHC) at v.

Algorithm BHS-torus-42:
The agent puts two tokens on its starting node (homebase). It then performs a Cautious-Walk in the East direction. If the agent survives, it returns, picks up one token and repeats the Cautious-Walk with one token in the East direction (leaving the other token on the homebase) until it reaches a node containing one or two tokens.

- If the agent reaches a node u containing two tokens, then the black hole is the next node on the East or on the South (See Property C of Proposition 4.1). The agent stops its exploration and checks whether the black hole is the East neighbor or the South neighbor.
- Whenever an agent reaches a node containing one token, it performs a Cautious-Walk in East direction with two tokens and then continues the Cautious-Walk in the same direction with one token. If the agent encounters three times[1] a node containing one token, it moves to the next horizontal ring below. To do that it first performs a Cautious-Walk with two tokens in the South direction. If the agent survives and reaches the ring below, it can start exploring this horizontal ring. If the current node is not empty, the agent repeats a cautious walk with two token in the East direction until it reaches an empty node. Now the agent repeats the same exploration process using the empty node as its new homebase. Whenever the agent encounter a node with two tokens, it stops its exploration and checks whether the black hole is the East or South neighbor.

[1]The agent may encounter homebases of two agents which have both fallen into the black hole. (In this case it must continue in the same direction until it locates the black hole)

Algorithm 2: BHS-Torus-42

```
   /* Algorithm for BHS in Oriented Torus (using k=4 or more agents, 2 tokens)                */
   /* One token = Homebase (or Blackhole in the East)                                         */
   /* Two tokens = BlackHole either in the East or in the South                               */
1
2  Found:= false;
3  while not Found do
4      Count := 0;
5      Put one token ;                                                  /* mark your homebase */
6      CautiousWalk(East,2);
7      repeat
8          if found single token then
9              increment Count; CautiousWalk(East,2);
10         else if found no tokens then
11             CautiousWalk(East,1);
12     until found two tokens or count = 3;
13     if found 2 tokens then
14         Found:=true;
15     else                        /* found 1 token (thrice), so move to the next horizontal ring */
16         Pick one token ;                                          /* Remove the homebase token */
17         CautiousWalk(South,2);
18         while found one token do                                   /* Search for an empty node */
19             CautiousWalk(East,2);
20         if found 2 tokens then Found:=true;
   /* The agent found two tokens and knows that one of the neighbors is the black hole        */
21 Go West;
22 Wait(1) ;                                                  /* To Synchronize with Cautious-Walk */
23 Go South;
24 if found two tokens then Mark-All(East);
25 else
26     if found one token then
27         CautiousWalk(East,2);
28     else
29         CautiousWalk(East,1);
30     Go North;
31     Mark-All(East);
```

In order to check if the black hole is the East neighbor v or the South neighbor w of the current node u (containing two tokens), the agent performs the following actions: The agent reaches the West neighbor x of w in exactly three time units by going west and south (and waiting one step in between). If there are two tokens on this node x then w is the black hole. Otherwise, the agent performs a cautious walk in the East direction with one token (or with two tokens if there is already one token on node x). If it safely reaches node w, then the black hole is the other node v. Otherwise the agent would have fallen into the black hole leaving a BHC at node w. An agent that discovers the black hole, traverses a cycle around the black hole visiting all its neighbors and marking all the links leading to it as dangerous.

Proposition 4.1 *During an execution of BHS-torus-42 with $k \geq 4$ agents, the following properties hold:*

A When an agent checks the number of tokens at a node, all surviving agents have picked up their cautious-walk token.

B At most three agents can enter the black hole:

(a) at most one agent going South leaving two tokens at the previous node.

(b) at most two agents going East, each leaving one of its tokens at the previous node.

C When an agent checks the number of tokens at a node, if there are two tokens then the black hole is either the East or the South neighbor of the current node.

D After an agent starts exploring a horizontal ring, one of the following eventually occurs:

(a) If this ring is safe, the agent eventually moves to the next horizontal ring below.

(b) Otherwise, either all agents on this ring fall into the black hole or one of these agents marks all links to the black hole.

Theorem 4.2 *Algorithm BHS-torus-42 correctly solves the black hole search problem with $k \geq 4$ agents, each having two tokens.*

Proof : Property B of Proposition 4.1 guarantees that at least one agent never enters the black hole. Property D ensures that one of the surviving agents will identify the black hole. Property C shows that if the links incident to a node w are marked as dangerous by the algorithm, then node w is the black hole. ■

5 Optimal algorithm for BHS using 3 agents and 2 tokens each

5.1 Sketch of the algorithm

Using the techniques presented so far, we now present the algorithm that uses exactly 3 agents and two tokens per agent. The algorithm is quite involved and we present here only the main ideas of the algorithm. The complete algorithm along with a proof of correctness can be found in Subsection 5.2 and 5.3.

Notice first that we can not prevent two of the three agents from falling into the black hole (see proof of Theorem 3.1). To ensure that no more than two agents enters the black hole, the algorithm should allow the agent to move only in two of the possible four directions (when visiting unexplored nodes). When exploring the first horizontal ring, an agent always moves in the East direction, using a Cautious-Walk as before and keeping one token on the starting node (homebase). This is called procedure *First-Ring*. Once an agent has completely explored one horizontal ring, it explores the ring below, using procedure *Next-Ring*. During procedure Next-Ring, an agent

traverses the already explored ring and at each node u of this ring, the agent puts one token, traverses one edge South (to check the node just below node u), and then immediately returns to node u and picks up the token. Note that an agent may fall into the black hole only when going South during procedure *Next-Ring* or when going East during procedure *First-Ring*. We ensure that at most one agent falls into the black hole from each of these two directions. The surviving agent must then identify the black hole from the tokens left by the other agents. For this algorithm, we redefine the *Black-Hole-Configuration* (BHC) as follows: If there is a node v such that there is one or two tokens at both the West neighbor of v and the North neighbor of v, then we say that a BHC exists at v. The algorithm should avoid forming a black hole configuration at any other node except the black hole. In particular, when the agents put tokens on their homebase, these tokens should not form a BHC. This requires coordination between any two agents that are operating close to each other (e.g. in adjacent rings of the torus) and it is not always possible to ensure that a BHC is never formed at a safe node.

The main idea of the algorithm is to make two agents meet whenever they are close to each other (this requires a complicated synchronization and checking procedure). If any two agents manage to gather at a node, we can easily solve BHS using the standard procedure for colocated agents[2] with the time-out mechanism (see [2, 4]) . On the other hand, if the agents are always far away from each other (i.e. more than a constant distance) then they do not interfere with the operations of each other until one of them falls into the black hole. The agents explore each ring, other than the first ring, using procedure *Next-Ring*. We have another procedure called *Init-Next-Ring* that is always executed at the beginning of *Next-Ring*, where the agents check for certain special configurations and take appropriate action. If the tokens on the potential homebases of two agents would form a BHC on a safe node, then we ensure two agents meet.

Synchronization:
During the algorithm, we ensure that two agents meet if they start the procedure *Init-Next-Ring* from nearby locations. We achieve this by keeping the agents synchronized to each other, in the sense that they start executing the procedure at the same time, in each iteration. More precisely, we ensure the following property:

Property 5.1 *When one agent starts procedure* Init-Next-Ring*, any other surviving agent either (i) starts procedure* Init-Next-Ring *at exactly the same time, or (ii) waits in its current homebase along with both its tokens during the time the other agent is executing the procedure or, (iii) has not placed any tokens at its homebase.*

Notice that if there are more than one agent initially located at distinct nodes within the same horizontal ring, an agent cannot distinguish its homebase from the homebase of another agent, and thus an agent would not know when to stop traversing this ring and go down to the next one. We get around this problem by making each agent traverse the ring a sufficient number of times to ensure that every node in this ring is explored at least once by this agent. To be more precise, each agent will traverse the ring until it has encountered a homebase node six times (recall that there can be either one, two or three agents on the same ring). This implies that in reality the agent may traverse the same ring either twice, or thrice, or six times. If either all agents start in distinct rings or if all start in the same ring then, the agents would always be synchronized with each other (i.e. each agent would start the next iteration of *Next-Ring* at the same time). The only problem occurs when two agents start on the same ring and the third agent is on a different

[2]Note that the agents meeting at a node can be assigned distinct identifiers since they would arrive from different directions.

ring. In this case, the first two agents will be twice as fast as the third agent. We introduce waiting periods appropriately to ensure that Property 5.1 holds.

For both the procedures *First-Ring* and *Next-Ring*, we define one *big-step* to be the period between when an agent arrives at a node v from the West with its token(s) and when it has left node v to the East with its token(s). During a big-step the agent would move to an unsafe node (East or South), come back to pick its token, wait for some time at v before moving to the next node with its token. The waiting period is adjusted so that an agent can execute the whole procedure *Init-Next-Ring* during this time. Thus, the actual number of time units for each big-step is a constant D which we call the *magic number*.

Algorithm *BHS-Torus-32*:

Procedure *First-Ring*:
During this procedure the agent explores the horizontal ring that contains its starting location. The agent puts one token on its homebase and uses the other token to perform cautious-walk in the direction East, until it enters a node with some tokens. If it finds a node with two tokens then the next node is the black hole. Thus, the agent has solved BHS. Otherwise, if the agent finds a node with a single token this is the homebase of some agent (maybe itself). The agent puts the second token on this node and continues the walk without any token (i.e. it imitates the cautious-walk). If it again encounters a node with a single token, then the next node is the black hole and the algorithm terminates. Otherwise, the agent keeps a counter initialized to one and increments the counter whenever it encounters a node containing two tokens. When the counter reaches a value of six, the procedure terminates. At this point the agent is on a node with two tokens (which it can use for the next procedure).

Unless an agent enters or locates the black hole, the procedure *First-Ring* requires exactly $6nD$ time units for an agent that is alone in the ring, $3nD$ for two agents that start on the same ring, and $2nD$ if all the three agents start on the same ring.

Procedure *Init-Next-Ring*:
An agent executes this procedure at the start of procedure *Next-Ring* in order to choose its new homebase for exploring the next ring. The general idea is that the agent checks the node u on the South of its current location, move its two tokens to the East, and then goes back to u. If there is another agent that has started *Next-Ring* on the West of u (i.e., without this Procedure, the homebases of the two agents would have formed a BHC), the agents can detect it, and *Init-Next-Ring* is designed in such a way that the two agents meet. More precisely, when an agent executes *Init-Next-Ring* on horizontal ring i without falling into the black hole, we ensure that either (i) it meets another agent, or (ii) it locates the black hole, or (iii) it detects that the black hole is on ring $i + 2$, or (iv) the token it leaves on its homebase does not form a BHC with a token on ring $i + 1$. In case (iii), the agent executes *Black-Hole-in-Next-Ring*; in case (iv), it continues the execution of *Next-Ring*.

Procedure *Next-Ring*:
The agent keeps one token on the homebase and with the other token performs a special cautious-walk during which it traverses the safe ring and at each node it puts a token, goes South to check the node below, returns back and moves the token to the East. The agent keeps a counter initialized to zero, which it increments whenever it sees a node with a token on the safe ring. When the agent sees a token on the safe ring, it does not go South, since this may be dangerous. Instead, the agent goes West and South, and if it does not see any token there, it puts a token

and goes East. If the agent enters the black hole, it has left a BHC. When the counter reaches a value of six, the procedure terminates.

During the procedure, an agent keeps track of how many (1 or 2) tokens it sees in the safe ring and the ring below. This information is stored as a sequence (of length at most 24). At the end of the procedure, using this sequence, an agent in the horizontal ring i can detect whether (i) the Black hole lies in the horizontal ring $i+1$ or $i+2$, or, (ii) there are two other agents in the ring i and ring $i+1$ respectively, or, (iii) the ring $i+1$ does not contain the black hole. In scenario (i), the agent executes procedure *Black-Hole-in-Next-Ring*; in scenario (ii), the agent meets with the other agent in the same ring; in scenario (iii), the agent moves to the next horizontal ring (i.e. ring $i+1$) to start procedure *Init-Next-Ring* again.

Procedure *Black-Hole-in-Next-Ring*:
The agent executes this procedure only when it is certain that the black hole lies in the ring below its current position. The procedure is similar to *Next-Ring*; the main difference being that the agent does not leave a homebase token. During the procedure, either (i) the agent detects the location of the black hole and marks all links to the black hole or (ii) the agent falls into the black hole, forming a BHC at the black hole.

5.2 Formal description of the algorithm

We now present in details the different procedures that are used in Algorithm 3. As explained before, the agents use procedure `FirstRing` to explore the horizontal ring where they start and `NextRing` to explore the others horizontal rings.

Algorithm 3: BHS-Torus-32

```
   /* Algorithm for BHS in Oriented Torus (using k=3 agents, 2 tokens)   */
 1 FirstRing;
 2 NextRing(true);
```

In the following, we sometimes denote nodes by their coordinates in the ring. The $North$ (resp. $East$, $South$, $West$) neighbor of node (i,j) is the node denoted by $(i-1,j)$ (resp. $(i,j+1)$, $(i+1,j)$, $(i,j-1)$).

Recall that for both the procedures *First-Ring* and *Next-Ring*, a big-step is the period between when an agent arrives at a node v with its token(s) to when it has left node v with its token(s). Note that in both procedures, a big-step takes the same number of time units that we denote by D.

FirstRing. During this procedure the agent explores the horizontal ring that contains its starting location. The agent puts one token on its homebase and uses the other token to perform cautious-walk in the direction East, until it enters a node with some tokens. This node is the homebase of some agent a (maybe itself). If there are two tokens on this node, then it means died on its first move going $East$. Thus, the agent has solved BHS. Otherwise, the agent puts the second token on this node and continues the walk using cautious-walk moves but without moving tokens. Note that, in this case, agent a has already explored these nodes, and so it is safe for the agent to continue going $East$ until it reaches a token. If it again encounters a node with a single token, then it must be the second token of agent a, because otherwise, a would have put its second token on top of this token. Thus, it implies that the next node is the black hole. Otherwise, the agent can only see nodes with two tokens on this ring. The agent keeps a counter initialized to one and increments the counter whenever it encounters a node containing two tokens. When the counter

reaches a value of six, the procedure terminates. At this point the agent is back on its homebase with its two tokens (which it can use for the next procedure).

When an agent locates the black hole, it marks all links incident to it, and then it uses the Procedure `CleanFirstRing` in order to remove all tokens that have been left on the homebases of the agents.

Unless an agent dies or locates the black hole, the procedure *First-Ring* requires exactly $6n$ big-steps (i.e., $6nD$ time units) for an agent that is alone in the ring, $3n$ big-steps (i.e., $3nD$ time units) for two agents that start on the same ring, and $2n$ big-steps (i.e., $2nD$ time units) if all the three agents start on the same ring.

Algorithm 4: FirstRing

```
   /* Algorithm for the first horizontal ring.  If two (or three) agents start in the
      horizontal ring where the black hole is located, the black hole is found.          */
 1 Put 2 tokens ;
 2 repeat
 3 |   reset clock;
 4 |   Go East;
 5 |   Go West;
 6 |   pick up a token;
 7 |   Go East;
 8 |   n := the number of tokens you see;
 9 |   if n = 2 then Mark-All(East), CleanFirstRing and Exit;      /* The black hole is found */
10 |   else Put a token;
11 |   Wait until clock reaches magic_number and reset clock;
12 until n > 0;
13 count := 1;
14 repeat
15 |   reset clock;
16 |   Go East;
17 |   if you see 1 token then Mark-All(East); CleanFirstRing and Exit; /* The black hole is found */
18 |   if you see 2 tokens then count := count + 1;
19 |   Wait until clock reaches magic_number and reset clock;
20 until count = 6;
21 Pick up 2 tokens ;
```

Procedure CleanFirstRing

```
   /* Procedure to use if the Black-Hole is in the first ring in order to remove all tokens
      from this ring except the one on the West of the Black-Hole                        */
 1 repeat
 2 |   Go West;
 3 |   if you see some tokens then pick them up;
 4 until until you see exactly one token or you meet an agent;
```

If two agents meet. If two agents meet at a node, then it is quite easy to locate the black hole using the team of two agents (without the help of any third agent). This algorithm is described below (see procedure *BHS-with-colocated-agents*). Notice first that it is always possible to break the symmetry between the agents who meet at a node, because the agents would arrive from different directions. In our algorithm, if two agents meet, they are both close from their two tokens and they first go to pick their tokens up. Once each agent has its two tokens, we do the following. After meeting, one of the agent becomes the leader and the other is the follower. Together they perform a combined cautious walk (Procedure *Cautious-Walk-With-Another-Agent*) described as

follows. The follower stays at the current node while the leader goes to the next node and returns immediately if the node is safe. Then both agents move together to the next node. The starting node is marked with three tokens (recall the agents have two tokens each, thus four tokens in total). The cautious-walk is repeated until the agents come back to the starting node[3]. The leader goes to the node directly below to check if this node is the black hole. If not, the agents now move to the next ring below along with the tokens, and repeat the whole process. Only the leader agent may fall into the black hole and in that case, the follower knows this fact within two time units, and thus it has located the black hole.

Procedure BHS-with-colocated-agents

```
   /* Procedure for BHS in Oriented Torus using 2 colocated agents,                    */
 1 Pick up any token;
 2 Go back to the home base of the first agent with both agents;
 3 Pick up any token;
 4 Go back to the home base of the second agent with both agents;
 5 Pick up any token;
 6 repeat
 7 |   Put 3 tokens;
 8 |   repeat
 9 |   |   Cautious-Walk-With-Another-Agent(East);
10 |   until three tokens found;
11 |   Pick up 3 tokens;
12 |   Cautious-Walk-With-Another-Agent(South);
13 until until black hole is found;
```

Procedure Cautious-Walk-with-another-agent(direction)

```
1 first agent moves to direction and go back;
2 second agent Wait(2);
3 if second agent does not see first agent then
4 |   Mark-All(direction) and Exit;
5 both agents move to direction
```

Remark 5.1 *In our algorithm, as soon as two agents meet, they execute* BHS-with-colocated-agents*. Note that, in this case, if the third agent sees tokens belonging to the two agents executing* BHS-with-colocated-agents*, it sees a tower of* 3 *tokens. Since in our algorithm, as long as no agents have met, there is at most two tokens on each node. Thus, the third agent can detect that the two other agents have met, and in this case, it stops the algorithm. We also assume that once two agents have met, if they meet the third agent while executing* BHS-with-colocated-agents*, then the third agent also stops executing the algorithm.*

Remark 5.2 *While executing the algorithm, if an agent visits a node which one of its incident links is marked as dangerous, then the agents stops executing the algorithm.*

NextRing. Once an agent has finished exploring the horizontal ring where it started executing the algorithm, it uses Procedure `NextRing` to explore the other rings. An agent executing `NextRing` on ring i knows that ring i is safe, and it wants to explore the nodes of ring $i+1$. To

[3]Note that there can be at most one node in the torus that contains three tokens.

Procedure NextRing(first_time)

```
   /* At any time during the execution, if you meet an agent you call procedure
      BHS-with-colocated-agents                                                    */
 1 reset clock;
 2 if not first_time then Go South else Wait(1);  /* to ensure all agents start InitNextRing at the
   same time.  */
 3 InitNextRing;
 4 count := 0; sequence := ε; danger := false;
 5 repeat
 6     Wait(12);
       /* You wait to enable an agent executing InitNextRing on the ring above to meet you if
          needed.                                                                  */
 7     if danger then
 8         Wait(2);
 9         danger := false;
10     else
11         Go South;
12         w := the number of tokens you see;
13         if w > 0 then
14             sequence := sequence ⊕ b_w;
15         Go North;
16     if count < 3 then Pick up 1 token;
17     Go East;
18     n := the number of tokens you see;
19     if count < 3 then Put 1 token;
20     if n > 0 then
21         count := count + 1;
22         sequence := sequence ⊕ t_n;
23         if count ≤ 3 then
24             if n = 2 or w = 2 then Mark-All(South) and Exit;        /* The black hole is found */
25             else if n = 1 and w = 1 then danger := true;
26             else
27                 Pick up 1 token;
28                 Go West, Go South;
29                 Put 1 token;
30                 Go East;   /* If you die, there is a token North and West of the Black Hole */
31                 Go West;
32                 Pick up 1 token;
33                 Go North, Go East;
34                 Put 1 token;
35         else
36             if w ≥ 1 then danger := true;
37     Wait until clock reaches magic_number and reset clock;
38 until count = 6;
39 Pick up all tokens;
40 Analyze(sequence);
```

do so, the agent first executes `InitNextRing` that we describe below. If an agent a continues executing `NextRing` after it has executed `InitNextRing`, we know that if a is on node (i, j), then there is no token on node $(i + 1, j - 1)$, and thus a can safely leave a token on (i, j) without creating a BHC with the node $(i + 1, j - 1)$. Then, the agent does a special cautious walk, i.e., it repeats the following moves until it meets a token on ring i: It leaves a token on its current node on ring i, goes $South$, goes $North$, picks up its token and goes $East$. If the agent dies, then it could have only died when it went $South$, and in this case, it has left a token on the node on the $North$ of the black hole. In case the agent safely reaches the node on the $South$, if it sees some tokens, it remembers how many tokens it sees.

When agent a has reached a node v on ring i where there is one or two tokens, a remembers how many tokens it sees. Either v is the homebase of some agent b (that may be the same as a), or the token (or the two tokens) on v indicates that the black hole is on the $South$ or on the $East$ of this node. However, if a is executing `NextRing` on ring i, it implies that ring i is safe and thus the black hole cannot be on ring i. Thus, either v is the homebase of an agent b, or the black hole is on the $South$ of v.

If there are two tokens on v, then v is the homebase of some agent b and the black hole is on the $South$ of v; in this case, a locates the black hole. If there is only one token on v, a cannot safely go $South$. However, we would like to check if the black hole is on the $South$ of v. To do so, a goes $West$ and then $South$ with its token; let u be the node reached (note that u is on the $South - West$ of v). If there is no token on u, a leaves its token, and then goes $East$. If the black hole is on the $South$ of v, a dies leaving a black hole configuration. If the black hole is not on the $South$ of v, a picks up its token and goes back to v.

If there is one or two tokens on u, it is a black hole configuration, and thus a cannot safely go $East$. If there are two tokens on u, then necessarily the black hole is on the $South$ of v, and a locates it. However, if there is one token on u, and one token on v, it does not necessarily means that the black hole is on the $South$ of v. Indeed, suppose that v is the homebase of a, that the black hole is on the $South$ of u, and that an agent c has started executing `NextRing` on ring $i + 1$ at the same time a started executing `NextRing` on ring i. After v has executed `InitNextRing`, there was no token on u, but c has died leaving its token on u later. Thus, if there is one token on v, and one token on v, a continues to execute `NextRing`.

If a has neither died, nor located the black hole at v, it continues to perform its special cautious walk until it sees some tokens on ring i twice. Each time it sees some tokens on ring $i + 1$, v remembers how many tokens it sees. Each time a sees some tokens on ring i, it remembers how many tokens it sees and checks if the black hole is on the $South$ as explained above.

Note that if the black hole is on ring $i+1$, it implies that v is not the homebase of a, and that another agent died leaving a token on $North$ of the black hole. Consequently, if v dies, it enters the black hole from the $East$.

Once a has seen three times some tokens on ring i, we can show that if the black hole is on ring i, either a died, or a located the black hole, or there is a BHC around the black hole. Thus, agent a leaves its second token on top of the token at its current node. Then it performs a special cautious walk, avoiding entering nodes marked by a BHC, until it sees tokens on ring i three more times. Again, while doing this, it remembers how many tokens it saw on nodes on rings i and $i + 1$. However, during this final traversal, whenever a reaches a node on ring i that contains one or two tokens, it does not check if the node on the $South$ is the black hole.

Remark 5.3 *In order to remember how many tokens it sees on the nodes of rings i and $i + 1$, the agent builds a sequence over the alphabet $\{b_1, b_2, t_1, t_2\}$. Initially, its sequence is empty; each time the agent sees one (resp. two) token on ring $i + 1$, it adds a b_1 (resp. b_2) to its sequence;*

each time the agent sees one (resp. two) token on ring i, it adds a t_1 (resp. t_2) to its sequence.

We can show that when an agent sees some tokens on ring $i+1$, these are either tokens left by dead agents, or homebase-tokens. This implies that the sequence is of length at most 24: an agent with finite memory can remember such a sequence.

In the description of the algorithm, the $\oplus$ symbol stands for the standard concatenation of string and ϵ for the empty string.

Procedure Analyze(sequence)

```
1  if sequence = b1t1b1t1b1t1b2t2b2t2b2t2 or sequence does not contain any b then
       /* Either you have seen no tokens on the ring below or you are a single agent that has
          seen tokens of another alive single agent                                          */
2      Go South;
3      NextRing (false);
4  else if sequence contains less than 3 t2 then
5      if you have only one token then
           /* In this case, the two other agents were in the same ring as you, they both died,
              and there is only one node in the ring containing two tokens: the node that is
              North of the black hole                                                        */
6          repeat
7              Go East
8          until you see two tokens;
9          Mark-All South and Exit;
10     else
           /* In this case, there was another agent in the ring with you, but you are the only
              one that is still alive                                                        */
11         BlackHoleInNextRing;
12 else if sequence contains two consecutive t then
       /* You know that there is another active agent in the ring, and that your sequence is
          different from the sequence of the other agent.                                    */
13     if sequence start with b then
14         Wait until you meet an agent
15     else
16         repeat
17             Go East
18         until you meet an agent;
19 else
       /* You know that the next ring is safe and that an agent dies exploring the ring below
          it.                                                                                */
20     Go South;
21     BlackHoleInNextRing;
```

How to use the sequence constructed during `NextRing`? At the end of Procedure `NextRing`, the agent a executing `NextRing` calls Procedure `Analyze`. This procedure enables an agent to distinguish which of the following cases happen (see Lemma 5.11).

- a does not see any tokens on ring $i+1$ and ring $i+1$ is safe; in this case a executes `NextRing` on ring $i+1$.
- there is no other agent on ring i and there is an agent that has executed `NextRing` without dying on ring $i+1$ when a was executing `NextRing` on ring i; in this case, a executes `NextRing` on ring $i+1$.

- there were two other agents executing `NextRing` on ring i, the black hole is on ring $i+1$, and a is the only agent that is still alive; in this case, a locates the black hole.
- there was another agent executing `NextRing` on ring i, the black hole is on ring $i+1$, and a is the only agent that is still alive; in this case, a executes `BlackHoleInNextRing` on ring i.
- there is another agent on ring i, ring $i+1$ is safe and the tokens the agents see on ring $i+1$ enable the two agents to meet.
- there is no other agent on ring i, black hole is in ring $i+2$, and there are the tokens of one or two dead agents on ring $i+1$; a executes `BlackHoleInNextRing` on ring $i+1$ with its two tokens.

InitNextRing. The aim of Procedure `InitNextRing` is to ensure that when an agent a start executing `NextRing` on ring i, the homebase of a does not form a BHC with a token on ring $i+1$.

In Figure 1, we have shown the different possible trajectories for an agent executing `InitNextRing`. The figure can be read as follows:

- the double-circled node is the place where the agent starts executing `InitNextRing`.
- the black node, if any, represents the black hole.
- the numbers in black on top of each node represent the time units where the agents arrive on the node.
- the numbers in red in the node represent the number of tokens belonging to other agents that the agent sees when it visits the node. If the agent visits the node twice and if it does not see the same number of tokens, we write x/y that means that it sees x tokens the first time and it sees y tokens the second time.
- the intervals in green below each node represent the nodes where the agent left its own tokens; note that any agent executing `InitNextRing` always moves its two tokens together. If an interval $[x-y]$ is written below a node, it means that the agent left its two tokens on this node between time units x and y. When an agent left its tokens on a node x time units after it started executing `InitNextRing` and that they are not moved before the end of `InitNextRing`, we write $[x-]$.

First, a leaves its two tokens on the node (i,j) where it starts `InitNextRing`, and then it goes *South* on node $(i+1,j)$ and remembers how many tokens it sees. Then the agent moves its two tokens *East* on node $(i,j+1)$. If the agent sees two tokens on node $(i,j+1)$, then it implies that the black hole is on node $(i+1,j+1)$ and agent a locates it.

Otherwise, if the agent has seen zero or two tokens on node $(i+1,j)$, it goes back to the node and checks how many tokens it sees. If a has seen zero tokens both times, we know that no agent has started executing `InitNextRing` at the same moment as a did. In this case, a enters node $(i+1,j+1)$ from the *North* (where it has left its two tokens) to meet an agent b that may be waiting there (if there is another agent in ring i, and if b is in the middle of the execution of `NextRing`); if it does not meet an agent, it goes back on node $(i,j+1)$.

If the agent has seen twice two tokens on node $(i+1,j)$, it means that the black hole is either on node $(i+1,j+1)$, or on node $(i+2,j)$. In this case, agent a enters node $(i+1,j+1)$ from the *North* (where it has left its two tokens); if a dies, it leaves a black hole configuration, and otherwise, the black hole is on node $(i+2,j)$ and a locates it.

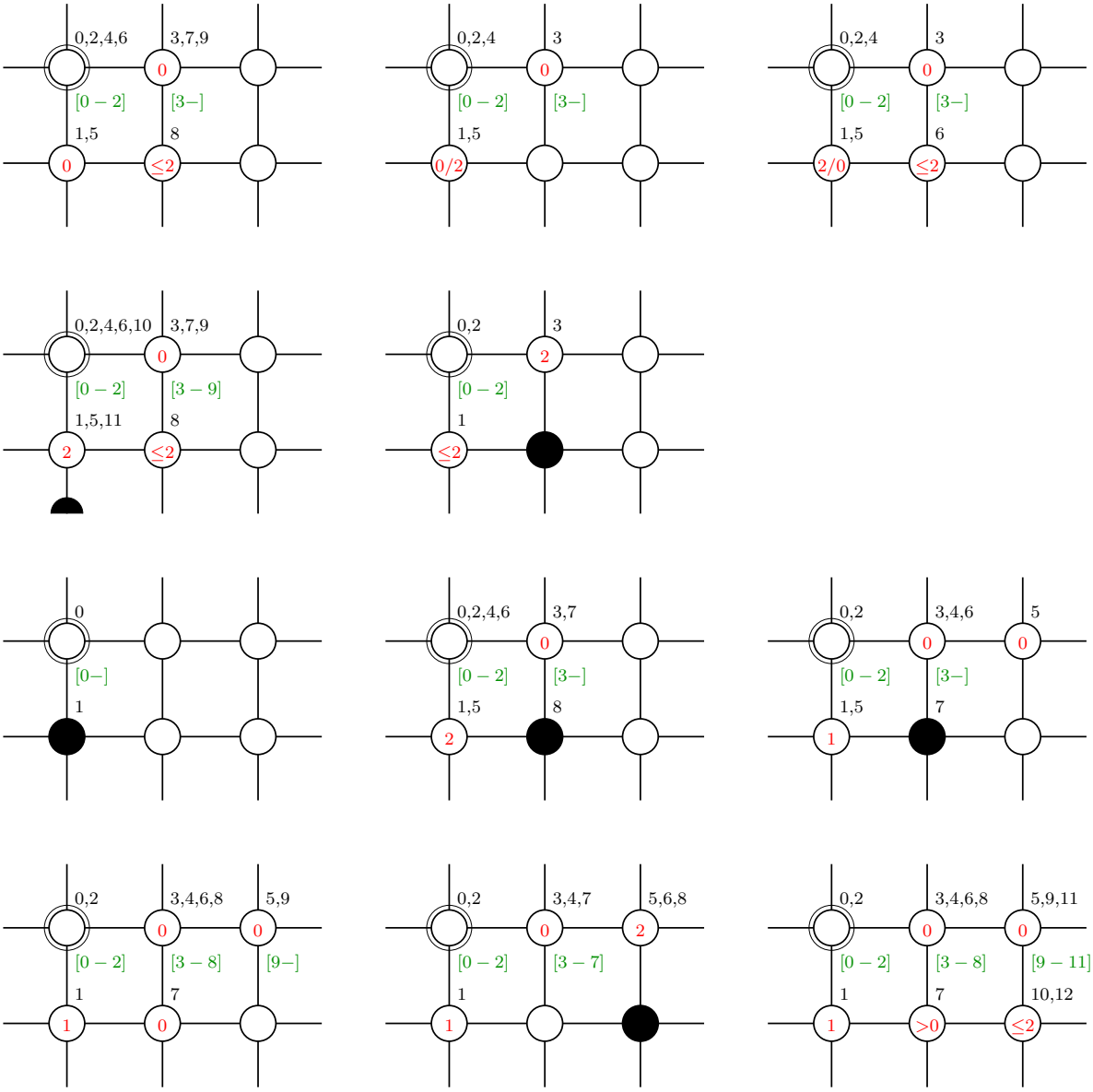

Figure 1: The different trajectories an agent can follow while executing `InitNextRing`.

If the agent has seen first zero (resp. two), and then two (resp. zero) tokens on node $(i+1, j)$, then it implies that another agent b has started executing `InitNextRing` on node $(i+1, j-1)$ (resp $(i+1, j)$) at the same moment as a did. In this case, a waits for b on the two tokens b left on node $(i+1, j)$ (resp. $(i+1, j+1)$).

Procedure InitNextRing

```
   /* Procedure used by the agent to proceed to the black hole search in the next ring.      */
   /* At any time during the execution, if you meet an agent you call procedure
      BHS-with-colocated-agents                                                              */
 1 Put 2 tokens;
 2 Go South;
 3 n1 := the number of tokens you see;
 4 Go North;
 5 Pick up 2 tokens;
 6 Go East;
 7 if you see 2 tokens then Mark-All(South) and Exit;
 8 else Put 2 tokens;
 9 if n1 = 1 then
10 |  OneTokenBelow;
11 else
12 |  Go West, Go South;
13 |  n3 := the number of tokens found;                          /* same place as for n1 */
14 |  switch the value of (n1, n3) do
15 |  |  case (0,0)
16 |  |  |  Go North, Go East;                                  /* back to your new homebase */
17 |  |  |  Go South, Go North; /* if you die, you leave 2 tokens on top of the Black-Hole */
18 |  |  |  Wait(3);          /* to ensure all agents finish InitNextRing at the same time */
19 |  |  case (0,2)
   |  |  |  /* You know that there is someone in the ring below you that did exactly the same
   |  |  |     thing as you did and it will come back to its homebase                        */
20 |  |  |  Wait until you meet an agent;
21 |  |  case (2,0)
   |  |  |  /* You know that there is someone in the ring below you that did exactly the same
   |  |  |     thing as you did and it will go back to its home base at East                 */
22 |  |  |  Go East;
23 |  |  |  Wait until you meet an agent;
24 |  |  case (2,2)
   |  |  |  /* An agent have died and have left two tokens next to the black hole           */
25 |  |  |  Go North, Go East, Go South;
   |  |  |  /* If you die, you leave the good configuration indicating the black hole and the
   |  |  |     third agent will find it                                                      */
26 |  |  |  Go North;
27 |  |  |  Pick up 2 tokens;
28 |  |  |  Go West;
29 |  |  |  Go South;
30 |  |  |  if no agent is waiting then Mark-All(South) and Exit;
```

The last case to consider is when agent a has seen one token on node $(i+1, j)$; in this case, agent a executes Procedure `OneTokenBelow`. Since the agents are synchronized, we can show that this token belongs to a dead agent b: the black hole is either in ring i, or in ring $i+1$. In this case, agent a first enters node $(i+1, j+1)$ from the $North$ (where it has left its two tokens); if a dies, it leaves a black hole configuration, and otherwise, it remembers how many tokens it sees on node $(i+1, j+1)$. If it does not see any token on node $(i+1, j+1)$, then a moves its two tokens on

node $(i, j+2)$: it knows that its homebase-token will not form a BHC with a node on ring $i+1$. If a sees at least one token on node $(i+1, j+1)$, it moves its two tokens on node $(i, j+2)$ and enters node $(i+1, j+2)$ from the *North*. If a died, it leaves a black hole configuration. Otherwise, it knows that the tokens a saw on nodes $(i+1, j)$ and $(i+1, j+1)$ are either homebase-tokens, or tokens indicating that the black hole is on ring $i+2$. However, if the black hole is on ring $i+1$, then it implies that both tokens a saw were homebase tokens, and thus two agents have executed `FirstRing` on ring $i+1$. But in this case, we know that the black hole has already been found and all homebase tokens of ring $i+1$ have been removed. Consequently, if a sees one token on node $(i+1, j)$, and at least one token on node $(i+1, j+1)$, then the black hole is in ring $i+2$. In this case, agent a executes `BlackHoleInNextRing` on ring $i+1$.

Procedure OneTokenBelow

```
   /* Procedure used by the agent that sees one token during InitNextRing. When the agent
      starts executing this procedure, it is one node east and one node north from the node
      containing one token.                                                              */
   /* At any time during the execution, if you meet an agent you call procedure
      BHS-with-colocated-agents                                                          */
 1 Wait(1);
 2 Go East;
 3 if you see 2 tokens then
 4 |   Wait(1);                        /* If an agent is doing the same thing, you wait for it. */
 5 |   Go West; Pick up 2 tokens; Go East;
 6 |   Mark-All(South) and Exit;
 7 else
 8 |   Go West, Go South;
   |   /* if you died you leave the good configuration indicating the black hole and the third
   |      agent will find it.                                                           */
 9 |   n2 := the number of tokens you see;
10 |   Go North;
11 |   Move 2 tokens to the East;
12 |   if n2 > 0 then
13 |   |   Go South;
   |   |   /* If you died you leave the good configuration indicating the black hole and the
   |   |      third agent will find it.                                                 */
14 |   |   Go North;
   |   |   /* There are two nodes with tokens in the ring below you and for both marked
   |   |      positions, the link to the East is safe. Thus, the ring below you is safe and
   |   |      the black hole is on the ring below this ring.                            */
15 |   |   Pick up 2 tokens;
16 |   |   Go South;
17 |   |   BlackHoleInNextRing;
18 |   else
19 |   |   Wait(3);    /* to ensure all agents finish OneTokenBelow and InitNextRing at the same
   |   |   time. */
   |   |
```

BlackHoleInNextRing. When it executes `InitNextRing`, or `NextRing`, an agent may locate the ring containing the black hole without locating the black hole itself. In this case, it executes the procedure `BlackHoleInNextRing`. When an agent a executes `BlackHoleInNextRing` on ring i, agent a has its two tokens, it knows that ring i is safe and that the black hole is on ring $i+1$. First, agent a reaches a node on ring i that does not contain any token; this ensures that agent is not on the *North* of the black hole. Then, agent a traverses the ring i until it finds a token on node (i, j). If there is a token on node $(i+1, j-1)$ (a can check this safely by moving *West* and then *South*), i.e., agent a discovers a black hole configuration, then we can show that the black

hole is on node (i, j), and agent a locates it. If there is no token on node $(i+1, j-1)$, a leaves its two tokens on node $(i+1, j-1)$ and enters node $(i+1, j)$ going *East*. If a dies, it leaves a black hole configuration. Otherwise, agent a repeat this procedure until it dies or locates the black hole. Since we know that there is a token on the node located on the *North* of the black hole, we know that either a locates the black hole, or a dies entering the black hole from the *East* leaving a black hole configuration.

Procedure BlackHoleInNextRing

```
   /* An agent executes this procedure when it knows that the ring it is moving in is safe, and
      that the Black Hole is in the ring below.                                              */
   /* At any time during the execution, if you meet an agent you call procedure
      BHS-with-colocated-agents                                                              */
 1 while you see some tokens do
 2     Go East;
 3 repeat
 4     repeat
 5         Go East;
 6     until you see some tokens;
       /* The Black Hole may be South; one needs to check this.                              */
 7     Go West;
 8     Go South;
 9     if you see some tokens then
10         Mark-All(East);
           /* You located the Black Hole since no agent had come back to pick the token.     */
11     else
12         Put 2 tokens;
13         Go East ;  /* If you die, there are some tokens North and West of the Black Hole. */
14         Go West;
15         Pick up 2 tokens;
16         Go North, Go East;
17 until you find the Black Hole;
```

5.3 Proof of Correctness for Algorithm BHS-Torus-32

Remark 5.4 *During the execution of our algorithm, once two agents have met, they execute* `BHS-with-co-located-agents`*, and they eventually find the black hole. We are mainly interested in showing that even if agents do not meet, our algorithm is correct.*

In a lot of the following lemmas, we implicitly assume that each agent that is still alive has not met any other agent.

Lemma 5.1 *Assume that when an agent starts executing* `InitNextRing` *on ring i, each other alive agent is either waiting with its two tokens on its homebases, or it starts executing* `InitNextRing`*, or it starts executing* `BlackHoleInNextRing`*. Moreover, assume that ring i is safe, and that on every node of ring i that contains tokens, there is an agent on its two tokens.*

When an agent a is the only agent on a node v, if it leaves v to visit a node w that may be the black hole, the following holds:

- *a always leaves v going East, or South,*
- *a always leaves ones or two tokens on v,*
- *all the tokens on v when a left belong to a,*

- *if w is not the black hole, the next move of a is to go back to v.*

Proof : The only times that an agent that is alone goes to a node that maybe the black hole are during the execution of `FirstRing`, `NextRing`, `BlackHoleInNextRing` or `InitNextRing`. When an agent executes `FirstRing`, it can only enter the black hole from the *East*, and this can only happen before it has seen any token. When an agent executes the main loop of `NextRing` on ring i, agent a can die from the *North* only before it meets a token on ring i. Thus, the only moves where a can enter the black hole is when it executes Line 4 of Procedure `FirstRing`, Lines 10 (when $count = 0$) and 29 of Procedure `NextRing`, Line 2, 17, 25 of Procedure `InitNextRing`, Lines 8 and 13 of `OneTokenBelow`, and Line 13 of Procedure `BlackHoleInNextRing`. One can check that in all these cases, all the properties hold. ■

Lemma 5.2 *If two or three agents start in the horizontal ring containing the black hole, then the black hole is found during the exploration of the first ring.*

Moreover, if two agents start in the horizontal ring containing the black hole, then the agent that locates the black hole removes all tokens left on the homebases of the agents before the third agent visits any node of this ring.

If one or more agents starts on a ring that does not contain the black hole, they finish executing `FirstRing` *on the vertex where they start without marking any link.*

Proof : Suppose first that two or three agents starts in the horizontal ring containing the black hole. Consider two agents a and b such that when moving *East* on the horizontal ring where the agents starts, a is the closer to the black hole, and b is the second closer.

Note that b cannot die while executing the loop between lines 2 and 11 of Procedure `FirstRing`, since it sees some tokens when it arrives on the homebase of a.

If a dies on its first move to the *East*, then a left two tokens on its homebase. Otherwise, a comes back and picks up a token before continuing its cautious walk, and thus b sees only one token when it arrives on the homebase of a. Consequently, the first time b sees some token, the black hole is located to the *East* of its current position if and only if b sees exactly two tokens.

Suppose the black hole is not located immediately on the *East* of the homebase of a. Agent a dies while executing the loop between lines 2 and 11 of Procedure `FirstRing` and leaves a token on the node on the *West* of the black hole. Thus, b first visits the homebase of a where it sees one token, and then visit a node with one token: it will mark the black hole links.

If there are three agents on the ring, let c be the third agent. Then it is easy to see that one of the following happens:

- either the third agent meets b while b is performing its cleaning phase, or once b has finished it,
- or c reaches the homebase of b before b has terminated its cleaning phase.

In the second case, c first visit a node with one token (the homebase of b), puts a token on top of it and then continue going *East*, and thus b picks up all the tokens that have been left on the homebases of a, b and c. If c arrives on the homebase of a before b has picked up the tokens, then c sees two tokens on the node and continue going *East* to reach the node where b marked the *East* link as leading to the black hole. If c arrives on the homebase of a once b has picked up the tokens, then after it has visited the homebase of b, it continues going *East* and reach the node where b marked the *East* link as leading to the black hole.

Suppose now that there is one or more agents starting in a ring that does not contain the black hole.

If there is only one agent in the ring, each time the agent sees some token, it is back on its homebase: the first time, there is only one token on its homebase and it adds a token. All the other times, it sees two tokens. Thus, the agent performs 6 turns of the ring during the execution of `FirstRing`.

If there are two agents a and b in the ring, the first time a sees some tokens, it is on the homebase of b and sees only one token. The next time it sees some token, it is back on its homebase, but b has arrived there before and left its second token on the node. The next time a sees some tokens, it is successively on the homebases of b, a, b, and a. Consequently, a is back on its homebase and has performed 3 turns of the ring during the execution of `FirstRing`.

When there are three agents a, b, c starting in the same ring, let assume that when we traverse the ring going *East* starting from the homebase of a, we reach the homebase of b before the homebase of c. For the same reasons as before, no agent has marked any link as leading to the black hole. And when agent a successively sees some tokens, it is successively on the homebases of b, c, a, b, c and a. Consequently a is back on its homebases when $count = 6$ and it has performed 2 turns of the ring during the execution of `FirstRing`.

When two agents start on the ring containing the black hole, it takes less than $2n$ big-steps to the surviving agent to locate the black hole and to remove tokens left on homebases. An agent that is alone in its ring needs $6n$ big-steps to finish executing `FirstRing` if the black hole is not in its ring. Thus, if two agents start on the ring i containing the black hole, the only token the third agent can see on ring i is the token located on the West of the black hole. Note that when the third agent reaches this node, the link going *East* has been marked as leading to the black hole and thus the agent stops executing the algorithm. ■

Lemma 5.3 *Consider two alive agents a and b and assume the black hole has not been found yet. When a starts the execution of* `InitNextRing` *on some ring i, either b is also starting the execution* `InitNextRing` *on some ring j, or b is starting the execution of* `BlackHoleInNextRing`*, or b is in the middle of the execution of* `NextRing`*, and is waiting on its homebase with its two tokens.*

Proof : We prove the lemma by induction on the numbers of rings explored by agent a, and we distinguish different cases.

First suppose that all agents have started in the same ring. If an agent dies during `FirstRing`, the black hole is found by the other agents. It takes exactly $2n$ big-steps (i.e., $2nD$ time units) to each agent to execute `FirstRing` or `NextRing` on each ring. Consequently, while no agent is dead, all agents always start the execution of `NextRing` simultaneously. Note that if an agent dies while executing `NextRing`, it enters the black hole from the $North$, and either another agent find the black hole when it sees the tokens of the dead agent, or it dies entering the black hole from the *East*.

Now assume that all agents have started in different rings. As long as no agent is dead, it takes exactly $6n$ big-steps (i.e., $6nD$ time units) to each agent to execute `FirstRing` or `NextRing` on each ring. Thus, if agent a and b are neither dead, nor executing `BlackHoleInNextRing`, they starts executing `NextRing` simultaneously.

The last case to consider is when two agents a and b start in the same ring, while the third agent c is in another ring. We also show that the three agents will never execute `NextRing` in the same ring.

While agents a and b are not in the same ring as c, it takes exactly $3n$ big-steps (i.e., $3nD$ time units) to agents a and b to execute `FirstRing` or `NextRing` on each ring, while it takes $6n$ big-steps to agent c. Consequently, as long as all agents are not in the same ring, each time

agent c starts `NextRing`, agents a and b start `NextRing` at the same time, unless they are dead, or executing `BlackHoleInNextRing`.

Suppose now that the three agents are not in the same ring and that a starts the execution of `NextRing`. If b is not dead, or executing `BlackHoleInNextRing`, b starts executing `NextRing` at the same time. Consider now agent c, and assume that it is still alive and that it is not executing `BlackHoleInNextRing`. Let q be the number of times a has executed `FirstRing` or `NextRing` so far, i.e., a has performed $3nq$ big-steps since it has started executing the algorithm. If q is even, then agent c has executed `FirstRing` or `NextRing` $q/2$ times, and starts executing `NextRing` at the same time as a. Otherwise, c has executed `FirstRing` or `NextRing` $(q-1)/2$ times and has performed $3n$ big-steps of `NextRing`. In this case, since c is alone in its ring, it means c is back in its homebase with its two tokens, and waiting for D time units.

We now show that the three agents cannot be in the same ring when they start executing `NextRing`. Since agent a and b are twice as fast as c, and since the agents start in two different rings, there will be a big-step where agent c is executing `NextRing` on ring $i+1$ while agents a and b are executing `NextRing` on ring i. Assume that agent c does not meet any other agent during `InitNextRing`. If c does not die while executing `NextRing` on ring $i+1$, then assume without loss of generality that agent a sees the token c left on its base before agent b. If agent c starts `NextRing` on ring $i+1$ at the same time as agents a and b start `NextRing` on ring i, agent c carries one of its token to perform a special cautious walk while it has left its other token on its homebase. If agent c is in the middle of executing `NextRing` on ring $i+1$ when agents a and b start `NextRing` on ring i, both tokens of agent c are on its homebase. In both cases, since the tokens of a and b always stay on ring i, agents a and b can see tokens on exactly one node of ring $i+1$.

Then the sequence a builds while executing `NextRing` starts with $b_1t_1t_1$ (or $b_2t_1t_1$), while the sequence of b starts with $t_1b_1t_1$ (or $t_1b_2t_1$). Due to the design of Procedure `Analyze`, this implies that agents a and b will not execute `NextRing` on ring $i+1$. ■

Lemma 5.4 *When an agent starts* `NextRing` *on ring i, it knows that ring i is safe, and that on every node of ring i that contains tokens, there is an agent on its two tokens.*

Proof : Consider an agent a that starts executing `NextRing` on ring i at time t. First assume that this is the first time agent a executes `NextRing`. From Lemma 5.2, we know that the black hole is not in ring i (otherwise, a is either dead or has located the black hole). In this case, we know that all agents in ring i are back on their homebases with their two tokens. Moreover, since the agents are synchronized, the only case to consider is when there are two agents b and c that started in ring $i-1$. While a executed `FirstRing`, agents b and c have executed `FirstRing` and a first iteration of `NextRing`. However, during the execution of `NextRing`, the only tokens b and c see on ring i are the two tokens of a on its homebase and thus the sequence computed by b and c are $b_2t_1t_1b_2t_1t_2b_2t_2t_2$ and $t_1b_2t_1t_1b_2t_2t_2b_2t_2$. In this case b and c have different sequences and they meet without leaving a unique token on ring i.

Assume now that agent a has already executed `NextRing` on ring $i-1$. Suppose that there exists a unique token on a node. Since all agents are synchronized, all alive agents are with their two tokens at time t (either before starting `NextRing`, or in the middle of the execution of `NextRing`). Thus, agent a has executed `NextRing` on ring $i-1$, and a has seen this token on ring i while executing `NextRing` on ring $i-1$. Thus, its sequence is different from $t_1^3t_2^3$. Moreover, if the sequence of a at the end of the execution of `NextRing` on ring $i-1$ is $(b_1t_1)^3(b_2t_2)^3$, it means that there are at least 6 other tokens on the ring i (three towers of 2) after it sees the token at position $(i, j+1)$. Since we have only three agents, this is impossible. ■

Lemma 5.5 *When executing* `InitNextRing` *on ring i, starting on node (i, j), if an agent sees some tokens on node $(i, j+1)$ without meeting an agent, then the black hole is located on node $(i+1, j+1)$, and the agent sees 2 tokens on this node.*

Proof : Consider an agent a executing `InitNextRing` in ring i, starting at position j and that sees some tokens at position $j+1$ on ring i (line 7 of Procedure `InitNextRing`).

We first prove that agent a sees two tokens on position $(i, j+1)$. Suppose that there is only one token on position $(i, j+1)$. Since the agent are synchronized, this token belongs to a dead agent b. Since an agent is moving its two tokens together during `InitNextRing`, agent b has died before a started `InitNextRing`. But, from Lemma 5.4, we know that this is impossible.

Thus, agent a sees two tokens at position $(i, j+1)$. If these tokens belong to a dead agent, then the black hole is either East or South of this node. Since agent a is performing `InitNextRing` on ring i, it knows ring i is safe, and thus, if there is a black hole, it has to be the South node.

Suppose these tokens belong to an agent b that is still alive. Since the agents are synchronized, either the agent is waiting on its homebase, or the agent has started the execution of `InitNextRing` at the same moment as a did. In the first case, the two agents meet. In the second case, it would mean that agent b and a started the execution of `InitNextRing` on the same node; which is impossible. ■

Consider an agent that starts executing `InitNextRing` on node (i, j). While executing `InitNextRing`, if the agent sees two (resp. zero) tokens the first time it goes to node $(i+1, j)$ and sees zero (resp. two) tokens the second time it visits node $(i+1, j)$, we say that the agent sees two tokens appearing (resp. disappearing).

Lemma 5.6 *During* `InitNextRing`*, if an agent sees tokens appearing or disappearing, then two agents meet, or the black hole is located.*

Proof : Suppose that an agent a executing `InitNextRing` on ring i at position j, sees 0 tokens (resp. 2 tokens) the first time it goes on ring $i+1$ at position j and 2 tokens (resp. 0 tokens) the second time it goes at position $(i+1, j)$. Since the agents are synchronized, all alive agents are either waiting at their homebases with their two tokens, or executing `InitNextRing`. Since the tokens have appeared (resp. disappeared), we know that there was an alive agent b that executes the lines 1 to 6 of Procedure `InitNextRing`. Consequently, we know that ring $i+1$ is safe and thus agent a can move East if the tokens have disappeared.

Thus, we know that 5 (resp. 6) time units after the beginning of the execution of `InitNextRing`, agent a is waiting on the two tokens located at the East of the homebase of agent b. These two tokens can be the two tokens of a dead agent, or the tokens of b that b moved during `InitNextRing`. In the first case, agent b locates the black hole (see Lemma 5.5).

Otherwise, if when going *South*, agent b sees either twice 0 tokens, or twice 2 tokens, then b is back on its two tokens 7 time units after the beginning of `InitNextRing`. If when going *South*, agent b sees 1 token, then b cannot die before the 8 time units after the beginning of `InitNextRing`. Moreover, either agent b meets the third agent at time 6 on the node located at the *East* of the node where a is waiting, or b is back on its two tokens at time 6 or 7. Thus the two agents meet.

Suppose now that agent b sees first 2 (resp. 0) tokens the first time it goes *South* and 0 (resp. 2) the second time. In this case, it means that there is an agent c executing `InitNextRing`. Consequently, we know that ring $i+2$ is safe. Thus, a, b and c are the three agents executing the algorithm, and they are located on lines i, $i+1$, $i+2$. If, when executing `InitNextRing`, agent c sees tokens appearing, or disappearing, then it means that these tokens are the tokens of agent a

(since there is only 3 agents executing the algorithm). Thus the torus has three horizontal lines and all of them are safe, which is impossible. Consequently, agent c does not see tokens appearing, or disappearing, while it executes `InitNextRing`, and thus agents b and c meet. ■

Lemma 5.7 *When executing* `InitNextRing` *on ring i at position j, if an agent sees twice two tokens on ring $i+1$ at position j, then one of the following holds:*

- *either, the black hole is on ring $i+1$ at position $j+1$, and the agent dies leaving a black hole configuration,*
- *or the black hole is on ring $i+2$ at position j and the agent locates it,*
- *or two agents meet.*

Proof : Suppose that an agent a executing `InitNextRing` on ring i at position j, sees 2 tokens the two times it goes on ring $i+1$ at position j.

There are two cases to consider: either the two tokens that a sees the first time have not been moved, or an agent b picked up these two tokens, while an agent c moved its two tokens to this node.

In the second case, it means that agents b and c are alive when agent a starts `InitNextRing`. Since the agents are synchronized, both agents are also executing `InitNextRing` and thus ring $i+1$ is safe. Since the three agents are located on ring i and $i+1$, agents b and c do not see any token on ring $i+2$ and since they both moved their tokens to the *East* of their starting positions, none of them died going *South*. Thus, they are back on their tokens 7 time units after the beginning of Procedure `InitNextRing`. Since ring $i+1$ is safe, agent a is back on ring $i+1$ at position j, 11 time units after it started executing Procedure `InitNextRing`: a meets another agent.

Suppose now that the two tokens seen by agent a the second time are the same as the two tokens it sees the first time. Since the agents are synchronized, all agents that are still alive are either waiting on their two tokens, or are executing `InitNextRing` and have picked up their tokens 2 time units after they started `InitNextRing`. If a does not meet another agent the first time it goes South, it implies that the agent that put these two tokens is dead. From Lemma 5.1, we know that the black hole is located either on ring $i+2$ at position j, or on ring $i+1$ at position $j+1$.

Assume that agent a does not meet any agent while it executes `InitNextRing`. If the black hole is located at position $j+1$ on ring $i+1$, agent a dies while going to this node, after it left its two tokens on ring i at position $j+1$. Thus, its two tokens and the two tokens on ring $i+1$ at position j form a black hole configuration. Otherwise, agent a does not die while executing `InitNextRing` and locates the black hole that is on ring $i+2$ at position j. ■

Lemma 5.8 *When executing* `InitNextRing` *in ring i, if an agent sees one token the first time it goes South, it knows another agent is dead, and one of the following holds.*

- *either it meets another agent,*
- *or it locates the black hole,*
- *or it dies leaving a black hole configuration,*
- *or it knows ring $i+1$ is safe and the black hole is in ring $i+2$, and executes* `BlackHoleInNextRing` *on ring $i+1$,*

- *or it continues executing* `NextRing` *without leaving a black hole configuration with a token on ring* $i+1$.

Proof : Consider an agent a executing `InitNextRing` on ring i, starting at position j. Since the agents are synchronized, when agent a executes `InitNextRing`, all alive agents are either waiting on their homebases, or executing `InitNextRing` and moving their two tokens together. Thus, if agent a sees a token on the node, then it is the token of a dead agent. This token may be the homebase token of the dead agent, or a token indicating that the black hole is located at the *South* or at the *East* of this token. In any case, the black hole is in ring $i+1$, or in ring $i+2$.

Claim 5.1 *If an agent b starts* `InitNextRing` *on ring $i-1$, at position j, $j+1$ or $j+2$, a and b meet.*

If an agent b starts at position j, or $j+1$, the claim follows from Lemma 5.6. Suppose now that an agent starts at position $j+2$ on ring i. From Lemma 5.4, we know that b cannot see a unique token on ring i at position $j+2$. Since ring i is safe, agent b does not see two tokens on ring $i-1$ at position $j+3$, and thus, agent b goes back at position $(i, j+2)$ 5 time units after it started `InitNextRing`. Since agent a is on position $(i, j+2)$ at this moment, the two agents meet.

Assume now that no agent starts `InitNextRing` on ring $i-1$, at position j, $j+1$, or $j+2$.

Claim 5.2 *If an agent b starts on ring i at position $j+1$, either the black hole is at position $(i+1, j+1)$ and a locates it, or a and b meet.*

If the black hole is at position $(i+1, j+1)$, then b dies leaving its two tokens on node $(i, j+1)$. From Lemma 5.5, a locates the black hole.

Suppose now that the black hole is not at position $(i+1, j+1)$. First note that since both a and b have their tokens with them when they start `InitNextRing`, and since there is one token on node $(i+1, j)$, from Lemma 5.5, agent b cannot see any token at position $(i, j+2)$. Moreover, b can either see 0 or 1 token at position $(i+1, j+1)$. If b does not see a unique token on node $(i+1, j+1)$, then b is at position $(i, j+1)$ 4 time units after it started `InitNextRing`; Since at this moment, a is also on this node, the two agents meet. Suppose now that b sees one token at position $(i+1, j+1)$. Since b cannot see any token on node $(i, j+3)$, b is at position $(i, j+2)$ 6 time units after it started `InitNextRing`. Since a sees 2 tokens on node $(i, j+2)$ 5 time units after it started `InitNextRing`, it waits there one time unit, and thus the two agents meet on this node.

Assume now that agent a does not meet any agent while executing `InitNextRing`.

Claim 5.3 *If agent a sees some tokens at position $(i, j+2)$, then a sees 2 tokens, the black hole is located at node $(i+1, j+2)$, and a locates it.*

For the same reasons as in the proof of Lemma 5.5, since agent a is executing `InitNextRing`, a can either see 0 or 2 tokens on node $(i, j+2)$.

Suppose the two tokens located at node $(i, j+2)$ belong to an agent b that is still alive. Since the agent are synchronized, either b is waiting on its two tokens, or agent b has started executing `InitNextRing` at the same moment a did. In the first case, a meets b. In the second case, agent b has picked up its two tokens and moved to the *East* 3 time units after it started `InitNextRing`. Consequently, it implies that agent b started on node $(i, j+1)$, but in this case, we know from the previous claim that a and b meet.

Thus, the two tokens a sees on node $(i, j+2)$ belong to a dead agent. From Lemma 5.1, the black hole is either located at the *South* or at the *East* of this node. Since a is executing `NextRing` on ring i, we know that ring i is safe, and thus the black hole is located at node $(i+1, j+2)$. Since a does not meet any other agent, it locates the black hole.

Assume now that agent a does not meet any other agent, and does not see any token on nodes $(i, j+1)$ and $(i, j+2)$.

Claim 5.4 *If the black hole is located on node $(i+1, j+1)$ or $(i+1, j+2)$, a dies leaving a black hole configuration.*

If the black hole is on node $(i+1, j+1)$, agent a dies 7 time units after it started `InitNextRing` leaving its two tokens on node $(i, j+1)$. With the unique token on node $(i+1, j)$, this leaves a black hole configuration.

If the black hole is on node $(i+1, j+2)$, then the token a sees on node $(i+1, j)$ is a homebase token of some dead agent b. From Lemma 5.1, b left a token on node $(i+1, j+1)$ that a sees at time 7. Thus, agent a dies at time 10 after it left its two tokens on node $(i, j+2)$. Consequently, with the token on node $(i+1, j+1)$, a dies leaving a black hole configuration.

Assume now that while it executes `InitNextRing`, agent a does not meet any other agent, and does not see any token on ring i. Furthermore, assume that the black hole is neither on node $(i+1, j+1)$, or $(i+1, j+2)$.

Claim 5.5 *If agent a sees one or two tokens on node $(i+1, j+1)$, then the black hole is in ring $i+2$, and ring $i+1$ is safe.*

We already know that the black hole is either on ring $i+1$, or in ring $i+2$. Since nodes located at position $(i+1, j+1)$ and $(i+1, j+2)$ are safe, we know that each of these tokens is either a homebase token, or a token indicating that the black hole is South.

Suppose that the black hole is on ring $i+1$. Then, the tokens left on both nodes are homebase tokens, and consequently either two agents have started in the ring and at least one of them found the black hole, or two agents died. If two agents have started executing the algorithm on ring $i+1$, then by Lemma 5.2, the black hole has been found, and while executing `NextRing` on ring $i-1$ (or `FirstRing` on ring i), agent a visited a node such that the link to the *South* was marked as leading to the black hole; i.e., a is not executing the algorithm any more. Otherwise, at least one agent has executed `NextRing` on node i before leaving its homebase on ring $i+1$, and thus it died leaving its homebase token on ring i, and not on ring $i+1$. Consequently, only one of the two tokens on positions $(i+1, j)$ and $(i+1, j+1)$ can be a homebase token, and consequently, the black hole is on ring $i+2$, and thus ring $i+1$ is safe. In this case, a executes `BlackHoleInNextRing` on ring $i+1$.

Claim 5.6 *If agent a continues the execution of* `NextRing`*, then its homebase token is not part of a black hole configuration with a token on ring $i+1$ when it terminates* `InitNextRing`*.*

If agent a continues the execution of `NextRing`, then it means that a does not meet any agent, does not see tokens on nodes $(i, j+1)$, $(i, j+2)$, or $(i+1, j+1)$, and that it leaves a homebase token on node $(i, j+2)$. Suppose that its token is part of a homebase configuration.

Suppose that an agent b leaves a token on node $(i+1, j+1)$ after agent a visited it. Then it means that agent b is still alive when agent a starts `InitNextRing`. Since the third agent is dead, b is either executing `BlackHoleInNextRing` on node $i+1$, or `InitNextRing` on ring $i+1$. In the first case, b does not leave any token on ring $i+1$. In the second case, the agent is executing

`InitNextRing` on ring $i+1$ and it means agent b has visited node $(i+1, j+1)$ ans has seen a unique token on the node; which is impossible from Lemma 5.5. ■

Lemma 5.9 *When executing* `InitNextRing` *in ring i, if an agent does not see any token and does not meet any agent, then it continues executing* `NextRing` *without leaving a black hole configuration with a token on ring $i+1$.*

Proof : Consider an agent a that does not see any token and does not meet any agent while executing `InitNextRing`. If agent a starts the execution on node (i, j), then it leaves its homebase token on node $(i, j+1)$, and there was no token on ring $(i+1, j)$ when agent a visited it. If a token appears on node $(i+1, j)$, then there is an agent b executing `InitNextRing` on ring $i+1$ that has started on node $j-1$ or $j-2$. In the first case, agent a has seen the two tokens on agent b the first time it went *South*. In the second case, agent b moves its two tokens to node $(i+1, j)$ only if it has seen a unique token on node $(i+2, j-1)$. But in this case, from Case 5.1 of Lemma 5.8, both agents have met. ■

In the following lemma, we show that when an agent executes `NextRing`, if it has not meet any other agent during `InitNextRing`, then there cannot be an agent that is located just below it. This implies that while executing `InitNextRing`, an agent cannot see the token another agent uses for its special cautious walk.

Lemma 5.10 *Once an agent has finished executing* `InitNextRing` *and continues the execution of* `NextRing` *(i.e., it is not dead, it has not meet any other agent, it is not executing* `BlackHoleInNextRing`*), it knows that there is no alive agent located immediately below it that executes* `NextRing`.

Proof : Consider an agent a that started executing `InitNextRing` on node (i, j) at time t. Once a has finished executing `InitNextRing`, it continues to execute `NextRing` only if it has seen no token on node $(i+1, j)$, or if it sees one token on node $(i+1, j)$ and no token on node $(i+1, j+1)$. In the first case, a is on node $(i, j+1)$. Since a did not see any token on node $(i+1, j)$ and did not meet any agent, we know from Lemma 5.6 that no agent has started executing `InitNextRing` on node $(i+1, j-1)$, or $(i+1, j)$ at time t. Thus, any agent that has started executing `InitNextRing` at time t cannot continue executing `NextRing` and be on node $(i+1, j+1)$. Since the agents are synchronized, we know that only an agent that was waiting on its homebase with its two tokens can be on node $(i+1, j+1)$; but in this case, agent a meets it at time $t+8$.

Suppose now that agent a has seen exactly one token on node $(i+1, j)$. In this case, a is on node $(i, j+2)$ when it has finished executing `InitNextRing`, and we know that the token on node $(i+1, j)$ belongs to an agent b that dies before a started executing `InitNextRing`. Assume there is an agent c that has finished executing `InitNextRing` on node $(i+1, j+2)$ and that neither c has met any other agent, nor c has located the black hole, nor c has started executing `BlackHoleInNextRing`. Note that this implies that c has started executing `InitNextRing` on ring $i+1$. If this is the first time a performs `InitNextRing`, then b died while performing `FirstRing` on ring $i+1$, and c also executed `FirstRing` on ring $i+1$. From Lemma 5.2, this implies that c locates the black hole while executing `FirstRing`. Consequently, a and c have already respectively executed `NextRing` on ring $i-1$ and on ring i. But in this case, c saw the token left by b on ring $i+1$ while executing `NextRing` on ring i. Thus the sequence computed by c contained at least one b_1. Since c has executed `NextRing` on ring i, its sequence should have been $(b_1 t_1)^3 (b_2 t_2)^3$. But this means that c saw at least 6 other tokens on the ring i (three towers of 2) after it sees the token at position $(i+1, j)$. Since we have only three agents, this is impossible. ■

Lemma 5.11 *When an agent a continue executing* `NextRing` *on ring i after it has finished* `InitNextRing`*, either a dies entering the black hole, or a locates the black hole, or a does not see any token on ring $i+1$, or a has seen some token on ring $i+1$ (i.e., sequence contains b_1 or b_2), and then a is in one of the following case and it can detect in which case it is if from the sequence it constructed.*

- *there were two other agents executing* `NextRing` *on ring i, the black hole is on ring $i+1$, and a is the only agent that is still alive; a locates the black hole.*
- *there was another agent executing* `NextRing` *on ring i, the black hole is on ring $i+1$, and a is the only agent that is still alive; a executes* `BlackHoleInNextRing` *on ring i.*
- *there is another agent on ring i, ring $i+1$ is safe and the tokens the agents see on ring $i+1$ enable the two agents to meet.*
- *there is no other agent on ring i and there is an agent that has executed* `NextRing` *without dying on ring $i+1$ when a has executed* `NextRing` *on ring i; a executes* `NextRing` *on ring $i+1$.*
- *there is no other agent on ring i, black hole is in ring $i+2$, and there are the tokens of one or two dead agents on ring $i+1$; a executes* `BlackHoleInNextRing` *on ring $i+1$ with its two tokens.*

Proof :

Consider an agent a that starts executing `NextRing` on ring i at time t; note that from Lemma 5.4, ring i is safe. If a does not see any token on ring $i+1$, and if ring $i+1$ is safe, a executes `NextRing` on ring $i+1$ once it has terminated executing `NextRing` on ring i.

Case 5.1 *The three agents start executing* `NextRing` *on ring i at time t and the black hole is on ring $i+1$.*

Let v be the node on the *North* of the black hole, and u the node on the *East* of the black hole. Without loss of generality, assume that c visits v before b, and that b visits v before a. Then, c dies leaving one or two tokens on v, and b dies leaving one token on u. When a arrives at v, if c left two tokens on v, b has located the black hole. Otherwise, v continues the execution of `NextRing` without entering the black hole (it has set its variable *danger* to *true*). During the execution of `NextRing` on ring i, a visits successively the homebase of b, the homebase of c, u, v, the homebase of a, the homebase of b, the homebase of c. It puts its second token on v when it arrives in v the first time, leaving a black hole configuration. Thus the sequence a has constructed is $t_1, t_1, t_1, b_1, t_1, t_1, t_1$. In this case, a goes to v (this is the only node with two tokens) and locates the black hole.

Case 5.2 *Ring $i+1$ is safe, two agents a and b start executing* `NextRing` *on ring i at time t.*

Note that a and b cannot die while executing `NextRing` on ring i.

First, assume that ring $i+2$ is also safe. Then the tokens a and b see on ring $i+1$ belong to agent c that is alive all along the execution of `NextRing` on ring i by agent a and b. Since rings $i+1$ and $i+2$ are safe, agent c is either executing `FirstRing` or `NextRing`. Suppose first that a and b are executing `NextRing` for the first time, i.e., they have just finished executing `FirstRing` and thus they have performed exactly $3n$ big-steps. During these first $3n$ big-steps c has visited its homebase three times and it has put a second token on its homebase when a and b start executing

`NextRing`. Moreover, it takes $3n$ big-steps to a and b to execute `NextRing`. During these $3n$ more big-steps, c visits its homebase 3 times and put a second token on its homebase at the end of these $3n$ big-steps. Consequently, during the whole execution of `NextRing` by a and b, c does not move its tokens. Suppose that c is in the middle of the execution of `NextRing` on ring $i+1$ when a and b start the execution of `NextRing` on ring i. Then c has just put two tokens on its homebase, and during the next $3n$ big-steps its tokens will not move. If c has started executing `NextRing` on ring $i+1$ when a and b start the execution of `NextRing` on ring $i+1$, then during the $3n$ big-steps it takes a and b to perform `NextRing` on ring i, c visits its homebase exactly three times and does not put a second token on its homebase before the moment a and b have finished executing `NextRing` on ring i.

Without loss of generality, assume that a visits the homebase of c before b. Consequently, when executing `NextRing`, a visits successively the homebases of c, b, a, c, b, a, c, b, a, while b visits successively the homebases of a, c, b, a, c, b, a, c, b. Each time a or b visits the homebase of c, they either always see one token, or always see two tokens.

The first three homebases on ring i that a or b sees contains one token, and both a and b add a token on the third homebase on ring i they see. Thus, the last third homebases a and b visit on ring i contain two tokens. Consequently, the sequence of a is $tb_1b_1tb_1b_2tb_2b_2$, while the sequence of b is $b_1tb_1b_1tb_2b_2tb_2$ where t is either t_1 or t_2. In this case, b waits on its homebase while a moves along ring i to meet b on its homebase.

Suppose now that the black hole is in ring $i+2$. Since the agents a and b some tokens on ring $i+1$, it implies that c has executed `NextRing` on ring $i+1$. We know that c has started executing `NextRing` on ring $i+1$ at time t or before. In any case, c is either dead before time t, or it dies during the first n big-steps of `NextRing`, i.e., before it comes back to its homebase. Once c is dead, there are two tokens left on ring $i+1$ (they may be both on the homebase of c if the black hole is the node below). Without loss of generality, assume that a visits the homebase of c before b. Let v be the node where is the second token of c, i.e., the node on top of the black hole.

If a visits v before it visits the homebase of b, then c is dead before a and b visit c and the sequence of a is $b_1^2t_1^2b_1^2t_1t_2b_1^2t_2^2$ (or $b_2t_1^2b_2t_1t_2b_2t_2^2$ if v is the homebase of c) while the sequence of b is $t_1b_1^2t_1^2b_1^2t_2^2b_1^2t_2$ (or $t_1b_2t_1^2b_2t_2^2b_2t_2$ if v is the homebase of c). In this case, b waits on its homebase, while a moves on ring i to meet b on its homebase.

If a visits v after it visits the homebase of b, then the sequence of a is $(b_1t_1)^3(b_1t_2)^3$. Note that if c has started `NextRing` on ring i at time t, the first time b visits v, c is not yet dead. Thus, the sequence of b is either $(b_1t_1)^3(b_1t_2)^3$, or $t_1(b_1t_1)^2(b_1t_2)^3$. In this case, both agents execute `BlackHoleInNextRing`.

Case 5.3 *The black hole is in ring $i+1$, two agents a and b start executing* `NextRing` *on ring i at time t.*

Let v be the node on the *North* of the black hole and assume without loss of generality that b visits v before a.

First suppose that no agent has explored ring $i+1$ yet. If b has started `NextRing` on v, b dies after its first move of `NextRing`. In that case, the first time a sees some tokens while executing `NextRing` is on v and it sees two tokens: a locates the black hole. Otherwise, a first sees the homebase token of b, and then the token b left on v. In this case, a dies entering the black hole from the West.

Let c be the third agent and assume that c has already executed `FirstRing` on ring $i+1$, or `NextRing` on ring i. If c has executed `NextRing` on ring i, it died leaving its tokens on ring i; but

in this case, a and b have executed `NextRing` on ring $i-1$ and they have seen the tokens c left. Thus, it implies that c died while executing `FirstRing`.

If c died on its first move, then it left its two tokens on its homebase, and thus b dies entering the black hole from v, but it leaves a black hole configuration, and a will locate the black hole while executing `NextRing`. Otherwise, let u be the node on the West of the black hole. Then b dies entering the black hole from v and a sees one token on u and one token on v and thus it does not enter the black hole. If we consider the tokens a sees on ring i while executing `NextRing`, a visits successively the homebase of b, v, the homebase of a, the homebase of b, v and the homebase of a. It adds a second token on its homebase the first time it visits it. Thus, the sequence a computed contained less than three t_2, and in this case, a will execute `BlackHoleInNextRing`.

Case 5.4 *Ring $i+1$ is safe, a is alone in its ring and ring $i+2$ is safe.*

Note that there cannot be two agents executing `NextRing` on ring $i+1$ while a is executing `NextRing` on ring i. Indeed, suppose a has executed `NextRing` q times, i.e., a has performed $6(q+1)n$ big-steps, then if two agents have started in the same ring, they have executed `FirstRing` once and `NextRing` $(12q+1)$ times, and thus they cannot be in the ring below a.

Since both rings $i+1$ and $i+2$ are safe, the tokens a sees on ring i are homebase tokens. If there is an agent b that has executed `NextRing` on ring $i+1$ while a is executing `NextRing` on ring i, then between two times a goes back to its homebase, it sees the base of b. Since the agents are synchronized, the third time a is on its homebase, b is also on its homebase and both of them put a second token on their homebases. Moreover, from previous lemmas, we know that the two homebases of a and b do not form a black hole configuration. Thus the sequence computed by a is $(b_1t_1)^3(b_2t_2)^3$. In this case, a executes `NextRing` on ring $i+1$.

Case 5.5 *The black hole is in ring $i+1$ and a is alone in its ring.*

If a is the first agent to explore ring $i+1$, a dies entering the black hole from the North.

If an agent b has explored ring $i+1$ before a, then b explored the ring using `FirstRing`, because otherwise, b would have left tokens on ring i, and a would have seen them while executing `NextRing` on ring $i-1$. In any case, a dies entering the black hole from the North.

If two agents b and c have already explored ring $i+1$, they have explored it using `FirstRing` and the black hole has been found.

Case 5.6 *Ring $i+1$ is safe, a is alone in its ring, and the black hole is in ring $i+2$.*

As before, we know it is not possible that two agents start executing `NextRing` on ring $i+1$ at time t.

Since a sees some tokens on ring $i+1$, we know that at least one agent has started executing `NextRing` on ring $i+1$ at time t or before.

First suppose that there is exactly one agent b that has started executing `NextRing` on ring $i+1$ before time t, i.e, b died while a was executing `NextRing` on ring $i-1$ or before. From the previous case, we know that b died entering the black hole from the $North$, leaving one or two tokens on the node v on $North$ of the black hole. If b died leaving its two tokens on v, then during the execution of `NextRing`, a visits v and its homebase six times, and the sequence computed by v is $(b_2t_1)^3(b_2t_2)^3$. If b died leaving only of of its tokens on v, a visits v, the homebase of b and its homebase six times, and the sequence it computes is $(b_1b_1t_1)^3(b_1b_1t_2)^3$. In any case, a executes `BlackHoleInNextRing` on ring $i+1$ once it has finished executing `NextRing` on ring i.

Suppose now that no agent executed `NextRing` on ring $i+1$ before time t, and that exactly one agent b starts executing `NextRing` on ring $i+1$ at time t. First suppose that b dies leaving two tokens on a vertex v; then it implies that the black hole is on the node on $South$ of v. Note that in this case, b dies while executing `InitNextRing`, and we know that if a starts `InitNextRing` on node (i,j) and b starts `InitNextRing` on node $(i+1,j-1)$, or $(i+1,j)$, a and b meet. Moreover, since b is dead when a finished `InitNextRing`, we know from previous lemmas that the two tokens of b and the homebase token of a do not form a black hole configuration. In this case, when executing `NextRing`, a never sees a black hole configuration, a visits six times v and its homebase, and the sequence it computes is $(b_2t_1)^3(b_2t_2)^3$. In this case, a executes `BlackHoleInNextRing` on ring $i+1$ once it has finished executing `NextRing` on ring i.

Suppose now that b does not die while it executes `InitNextRing`, i.e., when b dies, it has left one token on its homebase, and one token on the node v that is $North$ of the black hole. Note that the first time a visits v, b may still be alive (if a visits v before it visits the homebase of b), but the second time a reaches v, b is dead and a sees two tokens on v. During the execution of `NextRing`, a visits successively six times v, the homebase of b, and its homebase, or the homebase of b, v, and its homebase. In the first case, the sequence computed by a is $b_1t_1(b_1b_1t_1)^2(b_1b_1t_2)^3$ while in the second case, the sequence of a is $(b_1b_1t_1)^3(b_1b_1t_2)^3$. Moreover, from previous lemmas, we know that the homebase tokens of a and b do not form a black hole configuration. However it is possible that the homebase token of a and the token left on v form a black hole configuration. But in this case, there is one token on a, one token on v and thus, a does not visit the node u below its homebase and continues executing `NextRing`. Note that there cannot be a token on node u, because, it can only be the homebase token of b, but this is impossible from Lemma 5.10. Thus, a executes completely `NextRing` on ring i, and then it executes `BlackHoleInNextRing` on ring $i+1$.

Suppose now that two agents b and c have already `NextRing` on ring $i+1$, starting at time t or before. Since a sees some tokens on ring $i+1$, at least one of these two agents has executed `NextRing` on ring $i+1$. Again, we distinguish different cases: either b and c started executing `NextRing` on ring $i+1$ at the same time $t' \leq t$, or b has executed `NextRing` on ring i at time $t' < t$ while c has started executing `NextRing` on ring $i+1$ at time $t'' \leq t$.

Suppose that b and c started executing `NextRing` on ring $i+1$ at the same time $t' \leq t$. For the same reasons as before, we know that $t' < t$. From Case 5.3, we know that either the black hole has been found, or both agents are dead. If the black hole has been found, the first time a visits the node on $North$ of the black hole, it sees that the link going $South$ has been marked, and it stops the algorithm. If both agents are dead, we know that there is one token on ring $i+2$ that is on the node on the West of the black hole, and there are three tokens on ring $i+1$: the homebase of b, the homebase of c, and the node on the $North$ of the black hole. From Lemma 5.10, we know a cannot see a black hole configuration while executing `NextRing` on ring i. The sequence computed by a is then $(b_1^3t_1)^3(b_1^3t_2)^3$ and once a has finished executing `NextRing`, it executes `BlackHoleInNextRing` on ring $i+1$.

Suppose now that b has executed `NextRing` on ring i at time $t' < t$ while c has started executing `NextRing` on ring $i+1$ at time $t'' \leq t$. In this case c dies while executing `NextRing` on ring i, and we know that b start executing `BlackHoleInNextRing` on ring $i+1$ at time $t^* \leq t$. In this case, c has left either one or two tokens on the node v on the North of the black hole, and b does not leave any token on ring $i+1$, and dies leaving its two tokens on the node u on the West of the black hole, thus leaving a black hole configuration. In this case, the only tokens a sees on ring $i+1$ while executing `NextRing` on ring i are the homebase token of c and the token left by c on v. Since c is already dead when a starts executing `InitNextRing` on ring i, we know from previous lemmas that a cannot see any black hole configuration when executing `NextRing` on ring

i. Thus the sequence computed by a is $(b_1b_1t_1)^3(b_1b_1t_2)^3$, and a executes `BlackHoleInNextRing` on ring $i+1$ once it has finished executing `NextRing` on ring i. ■

An agent executing `BlackHoleInNextRing` first moves to find a place where there is no token. In the following lemma, we prove that in any ring there is always such a place.

Lemma 5.12 *If the black hole has not been found yet, when an agent a starts executing* `BlackHoleInNextRing` *on ring i, then there exists a node on ring i where there is no token.*

Proof :

Recall that an agent a is carrying its two tokens when it starts `BlackHoleInNextRing` on ring i and it knows that ring i is safe.

Note that on ring i, there can be at most 3 nodes containing tokens: two homebases, and one indicating the black hole is South. Thus if the size of ring i is at least 4 we are done.

Suppose now that the ring is of size 3. Since a is executing `BlackHoleInNextRing`, we know that either a has executed `NextRing` on ring i, or on ring $i-1$.

Suppose first that a has started executing `NextRing` on ring i at time t. Then we know from Lemma 5.11 that either the three agents have executed `NextRing` on ring i at time t, or that another agent b have executed `NextRing` on ring i at time t and that the third agent c was dead before time t. In the first case, since the ring is of size 3, one agent died on its first move when executing `InitNextRing`, and in this case, the black hole has been found from Lemma 5.5. In the second case, we know from the proof of Case 5.2 of Lemma 5.11 that c died while executing `FirstRing`, and thus its homebase token cannot be on ring i.

Consequently, a has executed `NextRing` on ring $i-1$ at time t. It implies that a saw tokens of one or two dead agents during `InitNextRing`, or during the execution of `NextRing`. Let b and c be the two dead agents and assume that c died before b. Since both b and c left their tokens on ring i, both b and c died while executing `NextRing` on ring i. First suppose that b and c did not execute `NextRing` on ring i simultaneously. Then, when c died, b has not finished executing `NextRing` on ring $i-1$, and from Lemma 5.11, b saw the tokens left by c, and b has not executed `NextRing` on ring i, but `BlackHoleInNextRing` on ring i. This implies that b did not leave its homebase token on ring i. Suppose now that b and c start simultaneously the execution of `NextRing` on ring i. If c died on its first move of `InitNextRing`, then from Lemma 5.5, b has located the black hole. Otherwise, since the ring is of size 3, c died leaving its two tokens on its new homebase, and then b locates the black hole when it arrives in c. ■

Lemma 5.13 *When an agent a executes* `BlackHoleInNextRing` *on ring i, at least one agent is dead and the black hole is on ring $i+1$, and either a locates the black hole, or a died entering the black hole leaving a black hole configuration.*

Proof : We know that if a executes `BlackHoleInNextRing` on ring i, it has executed `NextRing` on ring i, or on ring $i-1$, and we know from Lemmas 5.8 and 5.11 that ring i is safe, that the black hole is in ring $i+1$ and that a saw tokens of dead agents on ring i. Thus, at least one agent died while executing `NextRing` on ring i, and thus it left a token on top of the black hole.

Each time a sees a token on node (i,j), it goes to node $(i+1,j-1)$. If a does not see some tokens, it leaves its two tokens on the node an enter node $(i+1,j)$ from the *East*. Thus, if a dies, it leaves a black hole configuration. If a sees some tokens on node $(i+1,j-1)$, a marks all links leading to the node $(i+1,j)$ as leading to the black hole.

Since we know that there is a token on the node on the North of the black hole, we are sure that either a dies, or a marks links as leading to the black hole.

Suppose a is wrong when marking links going to node $(i+1, j)$ as leading to the black hole. Since we know the black hole is in ring $i+1$, this implies that the tokens on node (i, j) and $(i+1, j-1)$ are respectively the homebase tokens of two dead agents b and c. This implies that c died while executing `FirstRing` (otherwise its homebase token would be on ring $i+1$), and that b died while executing `NextRing`. Thus the token on node $(i+1, j-1)$ was already there when b executed `InitNextRing` on ring i. But from Lemmas 5.8 and 5.9, we know it is impossible. Thus, if a marks links leading to node $(i+1, j)$, the black hole is indeed in node $(i+1, j)$. ■

Lemma 5.14 *Suppose that two agents b and c are dead, and that the black hole is on node $(i+1, j)$. Then, there are one or two tokens on both nodes (i, j) and $(i+1, j-1)$. Moreover, the agent a that is still alive never enters the black hole and eventually locates it.*

Proof : Without loss of generality, assume that c died before b. We know that agents b and c died in one of the following way:

- c died executing `FirstRing` on ring $i+1$, and b died executing `NextRing` on ring i,
- b and c died executing `NextRing` on ring i,
- c died executing `NextRing` on ring i and b died executing `BlackHoleInNextRing` on ring i.

In the first case, the first agent let one or two tokens on node $(i+1, j-1)$ and the other agent let one or two tokens on node (i, j). In the second case, we know from Lemma 5.11 that it implies that the two agents were executing `NextRing` on ring i simultaneously. Thus, c died while performing its special cautious walk and left a token on node (i, j), and b died when during the execution of `NextRing` on ring i, it saw the token left by c and it enters node $(i+1, j)$ after it left a token on node $(i+1, j-1)$. In the last case, from the previous lemma, we know that b died leaving a black hole configuration.

If a is not executing `NextRing` on ring i with b when b dies, then we know that a executes eventually `NextRing` on ring $i-1$. When a executes `NextRing` on ring $i-1$, it sees some tokens on ring i, and from Lemmas 5.8 and 5.11, we know that a executes `BlackHoleInNextRing` on ring i. From the previous lemma, we know that a locates the black hole.

Suppose now that a is executing `NextRing` on ring i with b when b dies. Once b is dead, there are some tokens on nodes (i, j) and $(i+1, j-1)$. Thus, either a locates the black hole, or it continues to execute `NextRing` avoiding the node $(i+1, j)$ (its variable *danger* is *true*). We know from Lemma 5.11 that once a has finished executing `NextRing` on ring i, a executes `BlackHoleInNextRing` on ring i. From the previous lemma, a locates the black hole. ■

Theorem 5.1 *Algorithm* BHS-Torus-32 *correctly solves the BHS problem in any oriented torus with exactly three agents carrying two tokens each.*

Proof : We proved that if an agent reports it found the black hole, then it is always correct.

First, we know that if two or three agents start in the horizontal ring containing the black hole, then one agent locates the black hole.

Otherwise, the agents execute `NextRing` on consecutive rings until they see some tokens belonging to dead agents. Thus, at least one agent will die entering the black hole. Then, either a second agent locates the black hole, or it also dies entering the black hole, leaving a black hole configuration. In this last case, we know the third agent will not be killed and will eventually locates the black hole. ■

6 Conclusions

We showed that at least three agents are needed to solve BHS in oriented torus networks and these three agents must carry at least two movable tokens each for marking the nodes. The algorithm BHS-Torus-32 uses the smallest possible team of agents (i.e., 3) carrying the minimum number of tokens (i.e., 2) and thus, it is optimal in terms of resource requirements. However, on the downside this algorithm works only for $k = 3$ agents. In combination with algorithm BHS-Torus-42 (which solves the problem for any $k \geq 4$ agents carrying 2 tokens each), these algorithms can solve the black hole search problem for any $k \geq 3$, if the value of k is known. Unfortunately, algorithms BHS-Torus-32 and BHS-Torus-42 cannot be combined to give an algorithm for solving the BHS problem for any $k \geq 3$ agents without the knowledge of k: Algorithm BHS-Torus-32 for 3 agents will not correctly locate the black hole if the agents are more than 3, while in the algorithm BHS-Torus-42, 3 of the agents may fall into the black hole. Hence, whether the problem can be solved for $k \geq 3$ agents equipped with 2 tokens, without any knowledge of k, remains an interesting (and we believe challenging) open question. Another interesting open problem is to determine the minimum size of a team of agents carrying one token each, that can solve the BHS problem. Note that the impossibility result for three agents carrying one token each, does not immediately generalize to the case of 4 or more agents, as in those cases, we cannot exclude the possibility that two surviving agents manage to meet.

It is interesting to compare our results with the situation in a synchronous, oriented, anonymous ring, which can be seen as a one dimensional torus ([2]): The minimum trade-offs between the number of agents and the number of tokens, in this case, are 4 agents with 2 unmovable tokens or 3 agents with 1 movable token each. Additionally, in an *unoriented* ring the minimum trade-offs are 5 agents with 2 unmovable tokens or 3 agents with 1 movable token each whereas the situation in an unoriented torus has not been studied. Hence another open problem is solving the BHS problem in a d-dimensional torus, $d > 3$, as well as in other network topologies.

References

[1] M. A. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 75–85, 1994.

[2] J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for scattered black hole search in a ring. In *Proceedings of 18th International Colloquium on Structural Information and Communication Complexity*, lncs 6796, pages 186–197, 2011.

[3] J. Chalopin, S. Das, and N. Santoro. Rendezvous of mobile agents in unknown graphs with faulty links. In *Proceedings of 21st International Conference on Distributed Computing*, pages 108–122, 2007.

[4] C. Cooper, R. Klasing, and T. Radzik. Searching for black-hole faults in a network using multiple agents. In *Proceedings of 10th International Conference on Principles of Distributed Systems*, LNCS 4305, pages 320–332, 2006.

[5] C. Cooper, R. Klasing, and T. Radzik. Locating and repairing faults in a network with mobile agents. *Theoretical Computer Science*, 411(14-15):1638–1647, 2010.

[6] J. Czyzowicz, S. Dobrev, R. Kralovic, S. Miklik, and D. Pardubska. Black hole search in directed graphs. In *Proceedings of 16th International Colloquium on Structural Information and Communication Complexity*, pages 182–194, 2009.

[7] J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Complexity of searching for a black hole. *Fundamenta Informaticae*, 71(2,3):229–242, 2006.

[8] J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in synchronous tree networks. *Combinatorics, Probability & Computing*, 16(4):595–619, 2007.

[9] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.

[10] S. Dobrev, P. Flocchini, R. Kralovic, G. Prencipe, P. Ruzicka, and N. Santoro. Optimal search for a black hole in common interconnection networks. *Networks*, 47(2):61–71, 2006.

[11] S. Dobrev, P. Flocchini, R. Kralovic, and N. Santoro. Exploring a dangerous unknown graph using tokens. In *Proceedings of 5th IFIP International Conference on Theoretical Computer Science*, pages 131–150, 2006.

[12] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Multiple agents rendezvous in a ring in spite of a black hole. In *Proceedings of 6th International Conference on Principles of Distributed Systems*, pages 34–46, 2003.

[13] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: Optimal mobile agents protocols. *Distributed Computing*, 19(1):1–19, 2006.

[14] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48:67–90, 2007.

[15] S. Dobrev, P. Flocchini, and N. Santoro. Improved bounds for optimal black hole search in a network with a map. In *Proceedings of 10th International Colloquium on Structural Information and Communication Complexity*, pages 111–122, 2004.

[16] S. Dobrev, R. Kralovic, N. Santoro, and W. Shi. Black hole search in asynchronous rings using tokens. In *Proceedings of 6th Conference on Algorithms and Complexity*, pages 139–150, 2006.

[17] S. Dobrev, N. Santoro, and W. Shi. Scattered black hole search in an oriented ring using tokens. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, 2007.

[18] S. Dobrev, N. Santoro, and W. Shi. Using scattered mobile agents to locate a black hole in an un-oriented ring with tokens. *International Journal of Foundations of Computer Science*, 19(6):1355–1372, 2008.

[19] P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: Optimal black hole search with pure tokens. In *Proceedings of 22nd International Symposium on Distributed Computing*, LNCS 5218, pages 227–241, 2008.

[20] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *Proceedings of IEEE International Symposium on Parallel & Distributed Processing*, pages 1–10, 2009.

[21] P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48:166–177, 2006.

[22] P. Glaus. Locating a black hole without the knowledge of incoming link. In *Proceedings of 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pages 128–138, 2009.

[23] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Hardness and approximation results for black hole search in arbitrary graphs. *Theoretical Computer Science*, 384(2-3):201–221, 2007.

[24] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Approximation bounds for black hole search problems. *Networks*, 52(4):216–226, 2008.

[25] A. Kosowski, A. Navarra, and C. Pinotti. Synchronization helps robots to detect black holes in directed graphs. In *Proceedings of 13th International Conference on Principles of Distributed Systems*, pages 86–98, 2009.

[26] R. Kràlovic and S. Miklìk. Periodic data retrieval problem in rings containing a malicious host. In *Proceedings of 17th International Colloquium on Structural Information and Communication Complexity*, pages 156–167, 2010.

[27] E. Kranakis, D. Krizanc, and E. Markou. Deterministic symmetric rendezvous with tokens in a synchronous torus. *Discrete Applied Mathematics*, 159(9):896–923, 2011.

[28] C. E. Shannon. Presentation of a maze-solving machine. In *Proceedings of 8th Conference of the Josiah Macy Jr. Found. (Cybernetics)*, pages 173–180, 1951.

[29] W. Shi. Black hole search with tokens in interconnected networks. In *Proceedings of 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 670–682, 2009.